



Building an
Effective
Dev Portfolio

Josh Comeau

Building an Effective Dev Portfolio

Written by Joshua Comeau

© 2020 Joshua Comeau.

All rights reserved. The author has taken care in preparation of this book, but makes no expressed or implied warranty of any kind, and assumes no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in with or arising out of the use of the information contained herein.

This book is self-published.

Introduction.....	1
Motivation	2
Chapter I: Foundations	3
What is a developer portfolio?	3
Why do I need one?	3
Who is this book for?	4
Curating projects for your portfolio site.....	6
Projects not to include.....	8
How many projects to include?	10
What if my projects aren't good enough?	11
In summary	13
Chapter II: Strategy	14
Understanding the audience.....	14
A generic example	16
Things to avoid.....	24
In summary	27
Chapter III: Case Study	28
In summary	39
Chapter IV: Building.....	40
Design.....	40
Frameworks and tooling.....	42
Domain names.....	43
Hosting	45
Personalizing your portfolio	45
Accessibility.....	47
General layout	48
The "Project Detail" page:	48

Tone and copy	52
Developer blogs	54
In summary	55
Chapter V: Promoting Your Portfolio	56
Chapter VI: Cover Letters	58
Conclusion.....	60
A note on sharing	60
Appendix I: Portfolio Templates.....	61
Appendix II: Cover Letters.....	63
Letter 1: Khan Academy	63
Letter 2: Glitch	64
Letter 3: DigitalOcean	66

INTRODUCTION

According to Career Karma[†], about 34,000 people graduated from software-development bootcamp programs in 2019, in the US alone. This is in addition to all of the folks who self-taught through online programs like FreeCodeCamp, as well as folks graduating from college with Computer Science / Software Engineering degrees.

That's a lot of people competing for junior software development jobs!

In order to land an offer, you'll need to stand out from the crowd. A college degree from a prestigious university is one way to do this, but not everyone is privileged enough to be able to attend one. Prior work experience is another way, though there's a catch-22 here: if you need work experience to get a job, how can you possibly get work experience?!

Without a degree or previous work experience, **your portfolio** is the greatest asset you have. A couple of well-executed projects sends *such* a strong signal to potential employers. Portfolio projects show that you have the skills required to do the job.

We need to do some work to showcase those projects, though. Even the most amazing portfolio project might be overlooked if an employer only sees a couple screenshots and a Github link. We need to *guide* potential employers through our work, making sure to highlight the most interesting, impressive, salient parts. The good stuff. Stuff that isn't always obvious at first glance.

This book is an instruction manual for creating a portfolio site that stands out to prospective employers, in order to give you the best chance of landing an offer. It focuses on the practical tips and tricks I've learned over the years to help capture the attention of the folks who review applications.

[†] Source: <https://careerkarma.com/blog/bootcamp-market-report-2020/>

MOTIVATION

A few months ago, I offered to do volunteer portfolio reviews on Twitter, and got over 200 responses (<https://twitter.com/JoshWComeau/status/1227213590274486275>).

After reviewing a whole bunch of these portfolios, I realized I was giving the same feedback in 75%+ of cases. Most junior developer portfolios—even ones that were very well-built—were missing a lot of potential opportunities to stand out to employers.

I know about what employers are looking for because I've been involved in hiring for multiple organizations, including Khan Academy, Unsplash, Breather, and more. I've worked with HR departments to decide whether to interview someone or not. I've been on both sides of the table.

I've been using this information to help aspiring developers for years. I work with Concordia University (through a partnership with Journey Education) to help coach recent graduates and ensure that their applications are as polished and effective as possible.

For a while now, I've wanted to share this information more broadly. I had initially planned to write a blog post, but I quickly realized I had a lot to say about it! So instead, I've written a short book.

I'm passionate about this topic because I feel like it's so often underappreciated. While the portfolio site is just one piece of the job-hunting puzzle, it's a piece that can have an outsized impact. I don't want to oversell it—even the best portfolio site won't get you a job all on its own—but I've seen how a solid portfolio site can lead to more callbacks, more interviews, and ultimately more job offers. The goal of this book is to help you build a stellar portfolio that can increase your odds of success.

CHAPTER I: FOUNDATIONS

WHAT IS A DEVELOPER PORTFOLIO?

A developer portfolio is a website that showcases the work that you've done as a developer. The idea is borrowed from the art world: photographers, for example, will often create a portfolio website that showcases their best work.

The goal of a developer portfolio is to present yourself and your work in the best possible light. Your portfolio should make employers excited to meet you. It should offer a glimpse into how awesome it would be to work with you!

Unless you already have extensive work experience, your side projects are the greatest asset you have. They are your trophies, and the portfolio site is the trophy case that highlights all the amazing stuff you've done.

Building side projects is a lot of work, and we want to make sure to squeeze as much job-seeking value out of them!

WHY DO I NEED ONE?

You may be wondering if you actually *need* a developer portfolio. You have a resume, and a LinkedIn profile. You can list your projects there, isn't that enough? Will employers actually take the time to visit a portfolio site?

Without a doubt, resumes and LinkedIn profiles matter, but I believe that a developer portfolio can be a kind of *secret weapon*.

LinkedIn and resumes are both focused on *work* experience. If you're early in your career, you don't have very much relevant experience. And while LinkedIn does let you add details about individual projects, the format makes it very difficult to share all the context. As we'll see, presentation matters, and the presentation on LinkedIn is less than ideal.

Portfolio sites let us guide potential employers through our projects, making sure they realize how *freaking cool* the stuff we built is, and how *relevant* our skills are for the roles

they're trying to fill. They allow us to make a strong case for our competence and experience.

Now, not all employers will take the time to look at our portfolio site. And, believe it or not, **this is a blessing in disguise.**

Most developers either don't have a portfolio site, or they haven't put much thought or effort into one. The reason for this is that "build a solid portfolio site" is rarely a top priority for recent bootcamp grads, or self-taught folks. Many developers skip this step, since they see it as optional.

We can imagine an alternative reality, one in which every developer has a polished portfolio, and every employer scrutinizes them rigorously. In this reality, it would be very hard to stand out from the crowd. We'd have to put in a *bunch* of work just to meet the minimum expectation! It wouldn't give us an *advantage* at all.

Happily, we don't live in that reality. In ours, *many* employers will check a portfolio site, but not all. For those that do, we'll have a huge advantage.

Unfortunately, job-seeking is competitive by nature; for every junior dev job-posting, there will be multiple candidates, and only one of them will get an offer. A polished, effective portfolio site can be just the leg-up you need.

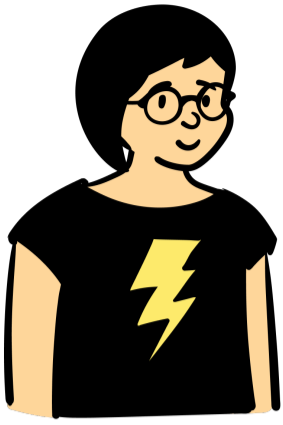
A portfolio site is a guided tour of your personal projects, a chance to show (rather than tell) that you have the skills needed for the job. Imagine if you're the only candidate to have a portfolio site listed with your application! I expect it'd be a serious competitive advantage.

Like I said: portfolio sites can be a secret weapon.

WHO IS THIS BOOK FOR?

I wrote this book for people who are trying to break into the tech industry, or who are looking to make early moves in their career. It assumes that you've been writing code for long enough to have built a couple projects, but not long enough to have an impressive work history. It assumes that you're looking to get a job working as a front-end or full-stack software engineer.

As I was writing this book, I kept 4 distinct personas in mind:



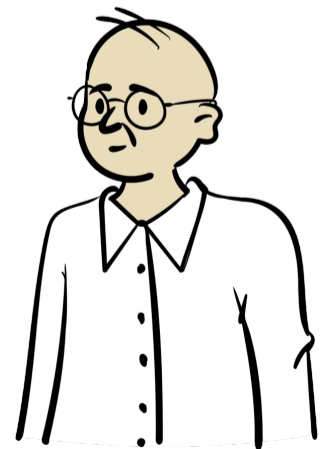
Morgan recently graduated from a 12-week coding bootcamp in Toronto. They have spent a few weeks polishing their final project from the course, and have started applying to positions.

Darius has been working as a junior front-end software developer for about 6 months in Austin. Unfortunately, he isn't enjoying his current role, and is looking to make a change. As he prepares to enter the job market again, he wants to give himself an edge by developing a solid portfolio site.



Priya is in her final year of Computer Science at UC Berkeley. She's participated in several hackathons and is particularly interested in full-stack development. She's contributed to several popular JS libraries. She's eager to get a job at a small start-up where she can have a large impact.

Levi is an office worker in Montreal who has been teaching himself web development through free online resources with React. It's been over a year of nights-and-weekend study, and he's even done a couple small freelance projects. To help with his office work, he created a small desktop application with Electron that lets him process invoices. He doesn't know if he's learned enough, but he's eager to change careers.



These people are all *early in their software development careers*. They all have a *body of work* they can share, and they're all hoping to *get a new job* working as a software developer.

I'm also focusing this book on front-end / full-stack software developers, but I expect that most of this content will also be relevant for back-end developers, mobile developers, or data engineers.

Caveats and Exceptions

- Software developers live and work all over the world, but I'm only familiar with hiring practices **in the United States and Canada**. If you're looking for a job in another part of the world, please understand that there may be differences that I am not aware of. Be skeptical of anything that doesn't sound right to you.
- This book assumes that you're **looking to get a job** as a software developer. If you're more interested in freelancing or starting your own company, the advice in this book is likely all wrong.
- If you don't yet have at least **1 solid personal project**, you should start with that. See the next section to get ideas for what to include.

CURATING PROJECTS FOR YOUR PORTFOLIO SITE

A portfolio site is a *showcase* of your work. But how do you know whether a project is worth showcasing? What sorts of projects qualify?

Ultimately, our goal is to show employers that we're competent at building stuff. There's a whole lot of different forms that this proof can take. Here are some examples, using our personas from earlier:

- **Morgan** spent 2 weeks at the end of their bootcamp building a restaurant reservation webapp, and they spent a few days afterwards tinkering and polishing it. The app is very much an MVP and it's missing a lot of features, but Morgan learned a lot building it.

Since graduating, Morgan has been volunteering at Code Dojo, a non-profit that runs bi-weekly workshops teaching programming fundamentals to kids. They've helped rework some of the curriculum.

- During the early stages of the Coronavirus pandemic, **Darius** spent a weekend building a Twitter bot using Ruby. It looks for people saying that coronavirus is “just like the flu”, and automatically replies with links to articles that clarify this misunderstanding.

Darius also spent 3 months in his first job contributing to a team that rolled out a notifications system.

- **Priya** has participated in several hackathons. For example, one time she teamed up with 3 other people to create an analytics platform that respects user privacy (A Google Analytics alternative with no GDPR banners!). She tackled the back-end side, ensuring that data is collected and processed effectively. The team completed a proof-of-concept, but have not taken any further steps to produce a real application.

Priya has also contributed to several JS libraries, like PancakeJS and em-dash.

- **Levi** created a small Electron app to help him scan and catalog invoices at his office job. It was never released to the public, since it's tailor-made for his specific use-case, but he uses it every day, and has saved him about 2 hours of work each week.

Levi also built a small single-page website for his local bakery, so that they could post their baking schedule, so that customers can always get fresh pies. 🥧

We're only scratching the surface — all kinds of different work qualifies. Morgan's volunteer work at a non-profit absolutely counts, even though there's no live demo or code samples. It still sends a strong signal about competence. Same thing applies for things like speaking at local meetups, being involved in the community, etc.

In my experience, certain kinds of projects are *especially* high-impact:

- 🌟 Did you build a project to solve a specific niche problem you had, like Levi's invoice-scanning app? Employers love to hear about projects like this, because it shows that you're able to apply your skills in creative ways, and tackle a project from start to finish.

- 🌍 Is your project “alive”? Is it shipped, and are people really using it for its intended purpose? It’s better to have a living, breathing project with real users (even if it’s only a handful) over a project that was built exclusively as a demo / for the portfolio.
- 🖥️ Have you contributed to a popular open-source library? Participating in the open-source community builds many of the “soft skills” that employers treasure! In order to get your change merged, you’ll need to be an effective communicator, be able to estimate tasks, and be good at receiving feedback.

If you’ve contributed to open-source, definitely include it as a project! Just be sure to be very explicit about what your contributions were. Non-technical contributions, like adding documentation, are super valuable, and absolutely worth adding to your portfolio, but be careful; you don’t want to imply that you’ve contributed in ways that you haven’t.

If you don’t have any projects that fit these criteria, that’s OK! This list is meant to help people decide how to prioritize existing projects. It isn’t meant to exclude other kinds of projects.

PROJECTS NOT TO INCLUDE

Tutorial / Workshop exercises

Don’t include projects that were created by following a step-by-step tutorial. Same thing goes for workshops or exercises assigned by a bootcamp.

There are a few reasons for this:

- Projects are meant to show how you can use the skills you’ve accrued to build things in a self-directed, unguided way. Tutorials are a way to give you those skills, but the tutorial itself is not a demonstration of those skills.
- If you try to pass a tutorial project off as your own, this can backfire if the screener recognizes the project (more likely than you’d think!).
- Projects should be an accurate representation of where your skills are at. If you get hired on the belief that a tutorial project was your own invention, you’ll be assigned

work that could be well beyond your current skill level. This is a stressful nightmare scenario!

If you made *significant* extensions to the project, it might be alright to include. For example, if you were given an assignment to create a Tic-Tac-Toe game, and you decided to spend a week adding online multiplayer support. Or if you transformed it into a Sudoku game. A good rule of thumb: did you spend >50% of the *total time* on the extensions? If the bulk of your time was spent adding self-directed extensions, then I think it's worth adding. Just be sure to be explicit about which parts were original.

Non-Dev Projects

In an attempt to “pad” the number of projects, I’ve seen some folks add information about non-development projects like photography or carpentry.

There is a place for these kinds of details on your portfolio, but it shouldn’t live amongst your other projects. You can include personal information in your About Me section.

Confidential Work

If you have previously worked as a junior dev / completed an internship, you may be tempted to include features that you worked on in that capacity. This can absolutely be a great idea, but you need to be careful and ensure that you’re able to talk about it. Especially if you work for an agency, you can get in trouble; agencies often do work under a non-disclosure agreement.

When in doubt, ask your employer for clarification.

The Portfolio Site Itself

Some developers will list their portfolio site as a project on their portfolio site. In general, I believe that it’s better not to do this; it gives the impression that you’re filling space because you don’t have other projects you can use.

The exception is if you invested time into parts of the site that are non-obvious from a user perspective. For example, did you build an “admin panel” to manage the content, like a mini home-built CMS? Did you code an analytics page to learn about traffic? Absolutely include it in this case!

HOW MANY PROJECTS TO INCLUDE?

At least 2. No more than 5.

Something which can be surprising: It's better to have 1 large polished project than 5 small projects.

You want to show that you can complete a non-trivial project from start to finish. Depth is more important than breadth, because it shows that you have the grit and determination to stick through the hard parts and finish a large project.

What if I have more than 5?

If you have a dozen projects that you feel are worth including, it's likely that you've gone *too wide and not deep enough*. Pick the 5 projects that best represent your skills and the work you want to do, and (if possible) consider spending a few days extending your largest project to be even larger.

Nobody will spend time looking at more than 5 projects. Most people will only look at 1 or 2. If you include 12 projects, a potential employer might only look at your least impressive one!

Portfolios are meant to highlight your *best* work, they aren't meant to be an exhaustive set of *all of your work*.

If you really want to include more than 5 projects, tuck the other projects behind a "show all" or "view archive" link. That way, you're setting a clear distinction between the "top-shelf" stuff and everything else.

What if I only have 1 project?

If you only have 1 significant project, you have a few options:

- Spend a weekend on a smaller second project. Consider building something relevant to your interests; if you're really into yoga, for example, build something that helps your

yoga practice. By picking something you're passionate about, you'll be more likely to want to work on it, and you'll wrap it up sooner.

- If you have an unfinished project, list it anyway, and note that it's "still under development".
- Is there a "spin-off" you could do of your first project? For example, if your first project was a full-stack web app that let users find food inspection results for local restaurants, could you maybe also create a Chrome Extension that spoke with the same server?
- Are there non-product things you've done that could be worth listing? Have you volunteered at any coding events? Have you contributed documentation to an open-source project? Think back through the time you've been learning to code, and see if anything pops up that could be used.

Ultimately, a portfolio site is only useful when you have projects to highlight. If you don't have at least 1 significant project and 1 minor project, consider spending some time filling in your portfolio. This can be an unfair burden on folks with caregiver responsibilities, or any other life situation that limits their time or energy. If it's unworkable, you might have to shift strategies; there are many ways into the industry! Unfortunately, this book's strategy assumes that you either already have suitable side projects, or else have the resources to create them.

WHAT IF MY PROJECTS AREN'T GOOD ENOUGH?

It's very common for junior developers to feel like their work isn't impressive enough to get a job. You probably aren't super confident in your skills, and you might look at your projects and feel like they're nothing special / nowhere near as good as they could be.

This is all a *common and normal part* of entering the industry. Most of the senior developers who have successful careers started out exactly where you are; worried that they're not experienced or knowledgeable enough to get a job.

In fact, if you've spent many hours on a project, there's a very good chance that you've faced and overcome significant technical challenges. Projects often have *hidden complexity*, and

employers are eager to hire developers who are good at solving problems. The main purpose of a portfolio site is to show prospective employers how you ran into technical challenges, and overcame them. Your projects don't need to be flashy or feature-rich.

This is why a detailed portfolio is so important. LinkedIn lets you add a screenshot and a couple paragraphs about a project, but think about how incomplete that story is. Most junior-developer projects look the same on the surface; the interesting stuff is *below* the surface. And a portfolio site is a great chance to guide the reader through that interesting stuff.

Of course, there *is* such a thing as a project which isn't complex enough to warrant being put on a portfolio site. But folks tend to overestimate the level of complexity which impresses employers, when framed correctly. Remember, you're going for junior-level positions! Reasonable employers will understand that you're not an expert yet (and you probably don't want to work for unreasonable employers anyway, so nevermind what they think).

If you wait until you feel ready to start working as a developer, you'll never start!

If you identify as female, there's a good chance that you're underestimating your own skills[†], and overestimating the level required for these positions. If you're feeling like you're not as advanced as some of the men in your peer group (eg. fellow bootcamp alumni, members of the same "learning to code" community), it's worth considering if maybe your peers are overestimating their abilities.

[†] Source: <https://www.fastcompany.com/40554829/youll-never-guess-which-gender-tends-to-overestimate-their-own-stem-skills>

IN SUMMARY

- Development portfolios can be a **secret weapon**, precisely because very few aspiring developers put thought and energy into them.
- You want to highlight your best work in your portfolio. Between 2 and 5 projects is ideal.
- All sorts of projects are suitable for this section, but you should avoid:
 - Work created by following a tutorial or completing an exercise
 - Projects that aren't related to software development
 - Confidential work
 - The portfolio site itself, *unless* you can tell a compelling story about it that isn't just summarizing what the visitor has already seen

CHAPTER II: STRATEGY

UNDERSTANDING THE AUDIENCE

Before we start sketching or coding, we need to think about who we're making our portfolio for, and what we want to communicate to them.

When you apply to work at a company, there are two types of people who are likely to visit your portfolio site as part of a screening process:

1. HR (Human Resources) hiring managers
2. Software developers

Both types of people are trying to answer the same question—whether you're worth hiring—but they have very different sets of criteria.

Our job will be to create a site that communicates our awesomeness to both groups of people. To do that, we have to learn a little bit about what they're looking for.

This section is a generalization / oversimplification. Every company has their own process. Sometimes, HR hiring managers are technical. Other times, you'll only ever be interviewed by full-time software developers. It's a helpful generalization because we need to plan for the common case, but it's important to never assume that a real person fits neatly into one of these two categories.

HR Hiring Managers

Unless you're applying to a very small startup, the company likely has an HR department. Depending on the size of the company, this department might have many employees who focus exclusively on hiring.

Generally, the first point of contact you'll have is with an HR person. This person likely isn't a software developer. Part of their job is to winnow the field; a typical junior developer job

posting might get up to 100+ applicants, and they'll want to reduce that number to something more manageable before scheduling interviews.

Here are some of the traits they're hoping we'll show in our application:

- **Competence.** Has this person built professional-looking projects before?
- **Enthusiasm.** Does this person seem eager to learn, excited for the kinds of work they would do on the job?
- **Fit.** Does this person have the right kind of personality for the company? Have they worked with the languages and frameworks we use here?

Critically, these folks are often very busy, since they have a lot of applicants to get through and many other job responsibilities to fulfill. They'll likely make a decision about moving forward in the process with you within 30-60 seconds. Our portfolio will need to capture their attention and communicate **competence, enthusiasm** and **fit** in a very short window of time.

Software Developers

It's important that our portfolio also has information that is relevant for software developers working at the company:

- At very early start-ups, your application will likely be screened by a developer, since they don't yet have an HR department
- Sometimes, HR hiring managers might reach out to a developer to help them evaluate a candidate during the screening process
- At all companies, you will almost certainly be interviewed by a software developer at the company, and this person may visit your portfolio beforehand, to learn about you.

The nice thing is that software developers were once in your exact shoes—they remember what it was like, searching for their first job!

Unlike HR hiring managers, developers generally aren't very interested in what languages or frameworks you know. In fact, for junior roles, developers care much more about your *potential* rather than what you happen to know already. They want to know if you'll be easy to mentor. They want to know if you have the grit to work through tough problems, and the humility to admit when you don't know something.

It can be hard to communicate this stuff through a portfolio site, but there are definitely things we should *avoid* doing. We'll discuss them more later in this chapter.

A GENERIC EXAMPLE

The main reason I'm writing this book is that, after reviewing hundreds of individual developer portfolios, I've realized that most junior developers aren't thinking strategically about how to stand out to these groups.

Let's take a look at a contrived portfolio I've built. As you review this portfolio, try and see it through the eyes of an HR professional, or a senior developer. Make some notes about what you like, what you don't like, and what's missing.

View it live: <https://generic-portfolio.now.sh/>

Squint at a screenshot, on the next page



**Full Stack Software developer
in Anytown, USA**

Looking for full-time opportunities

About me










I am a junior full stack software developer from Anytown. I am passionate about web development and about languages such as HTML, CSS, and Javascript.

I recently graduated from Apex College's 12-week intensive web development bootcamp and I am looking for a full-stack position.

When not coding, I enjoy hanging out with my friends.

Projects

Here are the projects I have completed:

 <p>Analytics app Application dashboard. Built with React, Redux, React Router.</p>  	 <p>Travel booking Website to book travel. Built with Vue.js, Vuex.</p>  	 <p>Marketing landing page School project. Built with vanilla HTML and CSS.</p>  
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Contact me

Let's work together!

Your name	Your email
<input type="text"/>	<input type="text"/>

Your message

Submit

Structurally, this portfolio has all the right pieces: it includes an “About me” section that tells the employer about the candidate, a list of projects they’ve built, and a contact form.

In fact, most of the portfolio sites I’ve reviewed have been similar to this one. If you’ve built a portfolio before, it’s likely quite similar to this!

The main reason we have a portfolio is to showcase our strengths. This portfolio isn’t really showcasing anything. It’s not helping the candidate at all; I suspect that if this candidate applied to 100 jobs, and included a link to this portfolio in 50 of those applications, there would be *no difference* in terms of response rate / number of booked interviews. This portfolio may as well not exist.

This is a perfectly acceptable generic portfolio, but we aren’t going for generic. We want to stand out, not blend in. This portfolio is missing opportunities to convey the stuff employers care about, like competence, enthusiasm, fit, and personality.

Let’s go over it in depth.

About Me

This is the “about” section on this portfolio:

About me

I am a junior full stack software developer from Anytown. I am passionate about web development and about languages such as HTML, CSS, and Javascript.

I recently graduated from Apex College's 12-week intensive web development bootcamp and I am looking for a full-stack position.

When not coding, I enjoy hanging out with my friends.

Here’s the biggest problem with this section: if you swap out the names of places and schools, *it can apply equally well to every single bootcamp graduate*. The only personal touch is the last line, and it’s hardly unique; everyone enjoys hanging out with their friends.

In terms of tone, it uses that “professional” tone people take when trying to get a job in the corporate world. I suspect that this tone might still be relevant when applying to work in a different industry, but it’s the wrong tone for the vast majority of tech roles.

What should an “About Me” section look like? We’ll discuss more in Chapter 4, but here’s a quick example of how I might write this for myself, when I was just starting out.

Hi there! I’m Josh. I’m a passionate introvert who loves building things with code.

My first experience with programming was when I was 12, but I didn’t last long: I was a big fan of computer-animated TV shows like Reboot, and I decided I wanted to do “computer graphics”. My mom bought me a C++ Reference Manual. It was several hundred pages, and totally inscrutable to me.

Years later, I decided to try learning Python. It was a radically different experience. I distinctly remember the moment it clicked; I was watching The Price is Right, and they were spinning the big wheel. A contestant landed on 0.70, and it made me wonder: statistically, what was their best move?

It occurred to me: I had the skills to build a simulation! So I threw together a Python script that would run thousands of simulations to work out what the right answer was. 15 minutes later, I had the answer. My burgeoning skillset was a superpower: I could derive answers that were previously off-limits to me. It was magic.

Since then, I’ve been honing my skills and learning Javascript. I recently graduated from Apex College’s 12-week intensive Web Development Bootcamp program. I’m seeking a full-time role where I can help a company achieve their goals.

This “About Me” section wouldn’t work for anyone else (well, unless they were willing to make up a big lie about their history!), and that’s the point. It’s true to who I am, and shares a compelling window into my personality and experience.

Your personality is likely different than mine, so your “About Me” section should be different too! You can be more formal, if that’s where your comfort zone is. You don’t have to share a quirky story about how you discovered programming. But it should be interesting, and it should open the door to more conversation. It should sound like something you might say in real life.

Let’s look at this through the lens of an HR hiring manager. You have a stack of applications to get through, and you can only realistically give 10-20% of them interviews. You click through to Charlie’s portfolio site, and you see the original generic “About me” section. You spend 3 seconds skimming it, and decide to skip to the next section, since there’s nothing of *interest* there. Unless something else on the page catches their eye, they’ll likely move straight on to the next applicant in the pool. By the time they get to the end, they’ll have forgotten yours entirely.

Compare that to the revised description, with the anecdotes. As their eyes skim the section, they catch “*The Price Is Right*”, which is a very unusual thing to spot on a portfolio site! They start reading from the beginning, and they get hooked in. They’re interested to know more about how you solved the problem, and what the answer was (should the contestant spin again??). You’ve created a memory, and a mild emotional response. After getting through the stack of applicants, it’s likely that yours will still be in the back of their mind. *Most* portfolio sites are generic, and it doesn’t take much to have yours stand out.

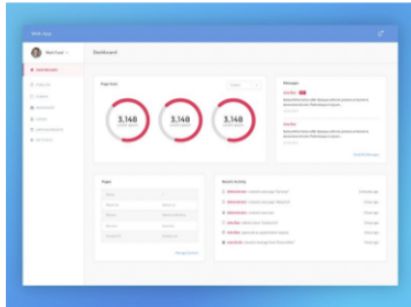
Hiring managers care about all the things you’d expect — work experience, volunteer experience, projects, education, skillset — but they’re also looking for people they’d enjoy working with. A quirky (and, critically, inoffensive) anecdote can endear you to them. Someone in that stack might have had a lot more experience than you, but if they forget that the person exists, they won’t get the job. You might.

Projects

In my generic portfolio example, the biggest missed opportunity is the projects section. Tragically, this is also the part most similar to common, real-world examples:

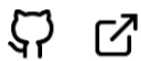
Projects

Here are the projects I have completed:



Analytics app

Application dashboard.
Built with React, Redux,
React Router.



Travel booking

Website to book travel.
Built with Vue.js, Vuex.



Marketing landing page

School project. Built with
vanilla HTML and CSS.



Each project is given the following information:

- A mini screenshot of the product
- A generic name
- 1 sentence description
- A list of technologies
- A link to the Github repo
- A link to a live demo of the product

You might feel like this is sufficient. After all, for those who are curious, they can follow the live link to get a demo, right? And software developers can follow the Github link to learn more about the implementation?

In truth, most people won't click on either icon. And those that do will likely come away unimpressed (or, at least, far less impressed than they should be!).

Be a tour guide

You don't want visitors to wander around on a live demo, or go spelunking in the codebase. We want to highlight the most impressive and interesting parts of our products!

With a live demo, you're taking a lot of risks:

- Have you tested your site on the browser and/or device the visitor is using?
- Will they follow exactly the same flow you've imagined? Unless you have a background as a senior UX designer, it's likely that they won't!
- Does your app require a signup, or any other high-friction setup? Most people don't have the patience to sign up for things.
- Have you found and fixed all possible edge-case bugs? Live demos going awry happens so much that it's cliché.

Even if everything goes flawlessly, think about what *isn't* conveyed in a live demo:

- Why is this project important to you? What inspired it? Why did you choose to build this?
- What are the major features that make it unique? How does it compare to existing products like it?
- What did you start with? Was this built from scratch? Did you have a team? If so, which parts did you do? Where did the design come from? Was there any collaboration?
- What was the hardest part of building this product? Where did you get stuck along the way?
- When you did get stuck, how did you resolve it? How did you overcome the obstacles you faced?

- What did you learn from doing this project? How has it affected the work you've done since then?

When it comes to the codebase, there are likely areas where you were able to come up with a clever solution to a hard problem, and other spots where you duct taped things together, or where things are working but you're not sure why. Every codebase has dusty closets. If a developer does check the link to your Github and start poking around, will they see the best bits?

Earlier, we spoke about how a portfolio lets us *squeeze value* out of our personal projects. We built a cool thing, but nobody knows about all the stuff that makes it compelling!

We want to **tell a story** about how a project came to be. It's much more interesting than showing a final product, without context. It also sends a much stronger signal to employers about what you can do!

It's probably still a good idea to include links to a live demo, mostly to serve as proof that you actually built the thing you're describing. But it should be de-emphasized. Most folks won't check. The people who do will likely take 10 seconds to say "yep, this is what they said it would be", and bail.

*What if your project isn't a tangible product, like volunteer experience?
Happily, the "be a tour guide" strategy is a natural fit for this kind of project.
Follow the same process, answering the list of questions, and include photos
when possible.*

Narration and Personality

You may be sensing a theme here: both the About Me section and the Projects section should have an element of storytelling. We don't just want to talk about where we are now, we want to share some history of how we got here.

Most folks didn't decide when they were 5 years old to become a programmer—we took a winding path that led us to where we are now. We want to capture that.

For bonus points, you can have fun with the details. I call them **sprinkles**—small UI flourishes that make your portfolio stand out, and become much more memorable. This can be things like animations, interactions, and surprising details.

We'll talk much more about this in the chapters ahead.

Design

This portfolio has a so-so design. It isn't hurting, but it isn't helping either. In Chapter 4, we'll talk more about how to improve the design of our portfolio sites without needing to become a great designer.

THINGS TO AVOID

Skill charts

Many developers will include a bar graph or some sort of data visualization showing their skill levels in various languages/frameworks:



Please, please don't do this. It's absolutely going to do more harm than good.

First, the numbers are meaningless. What does *85% Javascript* mean? Surely it can't mean that you know 85% of everything there is to know about the language; I've been writing JS since 2009 and I'm at maybe 50%.

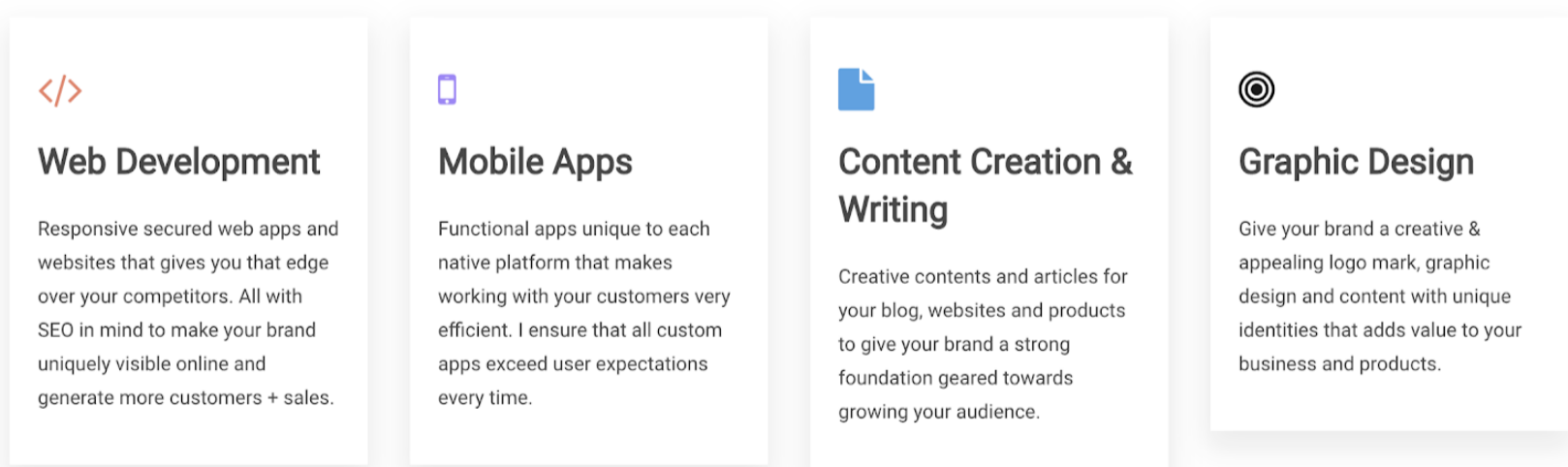
It's a no-win scenario. If the number you pick is too high to be realistic for someone of your level, you'll be seen as boastful and overconfident. If the number you pick is realistic, it will probably be low enough that it just doesn't look very good (who wants to hire someone with *20% Javascript*?)

These numbers are absolutely meaningless, and it's likely that the reader doesn't share your scale. I've never heard of one of these data visualizations helping a candidate, and I've seen them hurt the candidate quite often.

I know it can be tempting, because it can be a pretty fun element to build, from an engineering perspective. But it's a dangerous move. I advise skipping any sort of quantitative measure of your skill levels.

“What I Do”

Some portfolio sites have a “What I Do” section, which outlines a handful of high-level skills and abilities:



This sort of section is commonly seen on *freelancer* portfolios. If you're trying to start a consultancy or get short-term contract jobs, this can be a good way to highlight the services you offer.

This book is targeted at folks looking to get full-time software development jobs, though. In this case, it makes a bit less sense; you shouldn't be pitching your content-creation or graphic design skills if you're applying for a junior dev job.

While having complementary skills is a good thing, this sends a confusing message to prospective employers. It's a positioning problem; It makes it seem like you're not looking for full-time employment, because it fits a different mold.

I'd suggest maybe mentioning complementary skills on your resume, or in an interview. It doesn't need to be added to your portfolio site; save that real estate for your primary skillset.

Bravado

I've seen a few junior dev portfolios that include language like:

- *Head and shoulders above the rest*
- *Master of Javascript*
- *10x junior developer*

Most companies hire junior developers based on *potential*, not based on your current level of expertise. In fact, when a senior developer reviews your portfolio, they're hoping to see that you're eager to learn, humble, and looking for guidance; folks like that tend to be easier to mentor, and make for better teammates.

The problem with this language is that it gives the impression that you're finished learning. If you've mastered Javascript, how are you going to react when you submit some code to be reviewed, and a lot of changes are requested?

It gives the impression that you'll be hard to work with.

This is admittedly a rare problem; most folks *underestimate* their abilities, not *overestimate* them. But this still does happen sometimes.

IN SUMMARY

- We want to target our portfolio at two cohorts of individuals:
 - **HR Hiring Managers** are looking for competence, enthusiasm, and fit
 - **Software developers** are looking for folks with grit and determination, and who will be easy to mentor
- Structurally, our portfolio should feature 3 things on the homepage: an “About Me” section, a list of projects, and a contact form (or other contact method).
- In terms of tone, your portfolio site should reflect your personality. It should be specific enough that it would obviously be plagiarism if someone else tried to use your “About me” section. Avoid generic platitudes or corporate-speak.
- For your “Projects” section, live demos are overhyped and underutilized. We want to be a tour guide, guiding readers through the most impressive parts of our projects. **Tell a story.** We’ll learn more about how to do this in the coming chapters (this is arguably the most important part of this book!)
- Be creative with **sprinkles**—small bits of flair that showcase your enthusiasm for development. This is especially relevant for folks who are primarily interested in front-end roles.
- **Avoid skill charts**, or any other quantitative representation of your skills.
- **Avoid bravado.** You want to make it clear that you’re eager to learn from more experienced developers, not that you think you already know everything.

CHAPTER III: CASE STUDY

Of the hundreds of portfolios I have reviewed, the best one I've seen is by Julia Johnson, a student at Purdue University, and an intern at IBM.

With Julia's permission, I've used her portfolio in this book as a case study. Let's look through some screenshots. As you read through it, keep in mind all the stuff we've been talking about. Try to see it through the eyes of a company looking to hire.

You can view these screenshots online at:

- <http://joshwcomeau.com/images/effective-portfolio/julia-home.jpg>
- <http://joshwcomeau.com/images/effective-portfolio/julia-project.jpg>



Hi there, I'm Julia.

Front End **Developer**,

Student and Minimalist



I'm a 20 year old student at Purdue University currently studying web development and design.

As I've grown as a developer, I've worked alongside senior designers and developers who have raised my standards for what's expected of any web application.

Through these experiences, I've had the opportunity to create memorable products that are not only enjoyable to use, but are written in code that's maintainable and easy to understand.

My Skills

Apart from the courses included in my degree, I've taken a number of online courses such as The Complete Javascript Course, Advanced CSS & Sass, React: the Complete Guide, Javascript 30, and I'm currently taking ES6 for Everyone

◦ HTML5

◦ Javascript ES6

◦ React Native

◦ Git

◦ CSS3

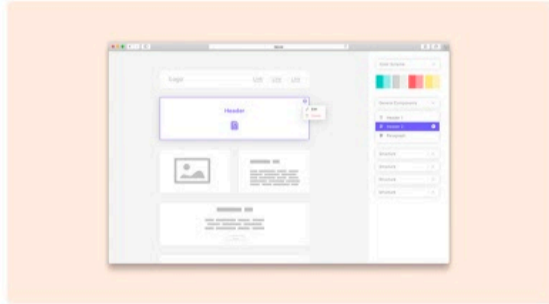
◦ React

◦ Styled-Components

◦ PHP & SQL

What I've been working on

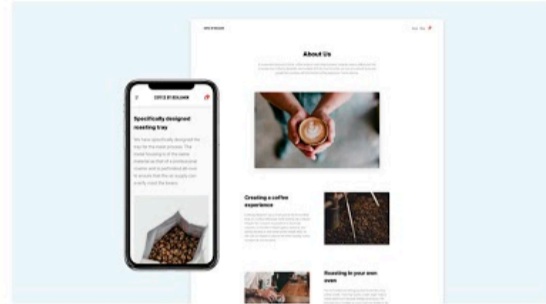
I like to stay busy and always have a project in the works. Take a look at some of the applications, articles and companies I've dedicated my time to.



Decore

A web application that builds custom starter layouts for developers

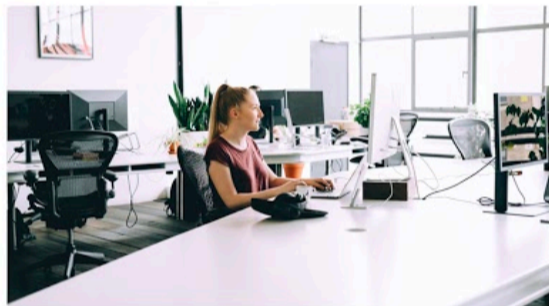
[VIEW PROJECT >](#)



Coffee By Benjamin

Developed a shopify e-commerce application with React for a coffee roasting company

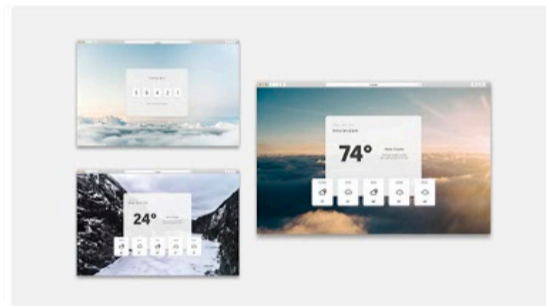
[VIEW PROJECT >](#)



Working at Awkward

Read about my experience as an intern at Awkward, a digital design agency.

[READ STORY >](#)

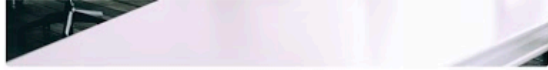


Forecast

A 7 day weather application that sets weather data based on user input.

[READ EXPLANATION >](#)

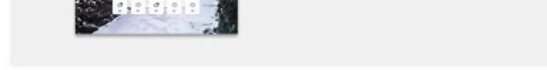




Working at Awkward

Read about my experience as an intern at Awkward, a digital design agency.

[READ STORY >](#)



Forecast

A 7 day weather application that sets weather data based on user input.

[READ EXPLANATION >](#)



Coffee Chemistry

My current work in progress, a web application that helps you brew the perfect cup of coffee.

COMING SOON

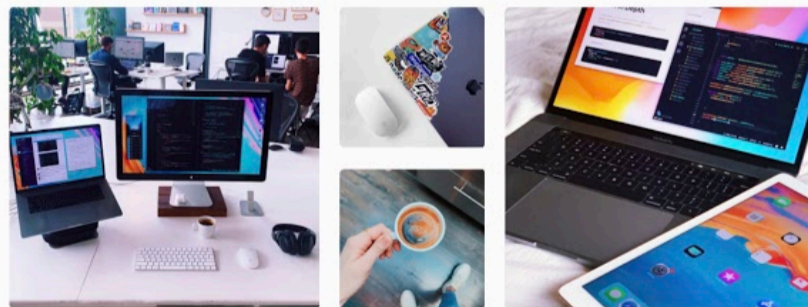
Let's Build Something Together

Feel free to reach out if you're looking for a developer, have a question, or just want to connect.

juliajohnson@purdue.edu

I'm a lot cooler on instagram

[SEE MORE >](#)



JU
LIA.

[github](#) [instagram](#)

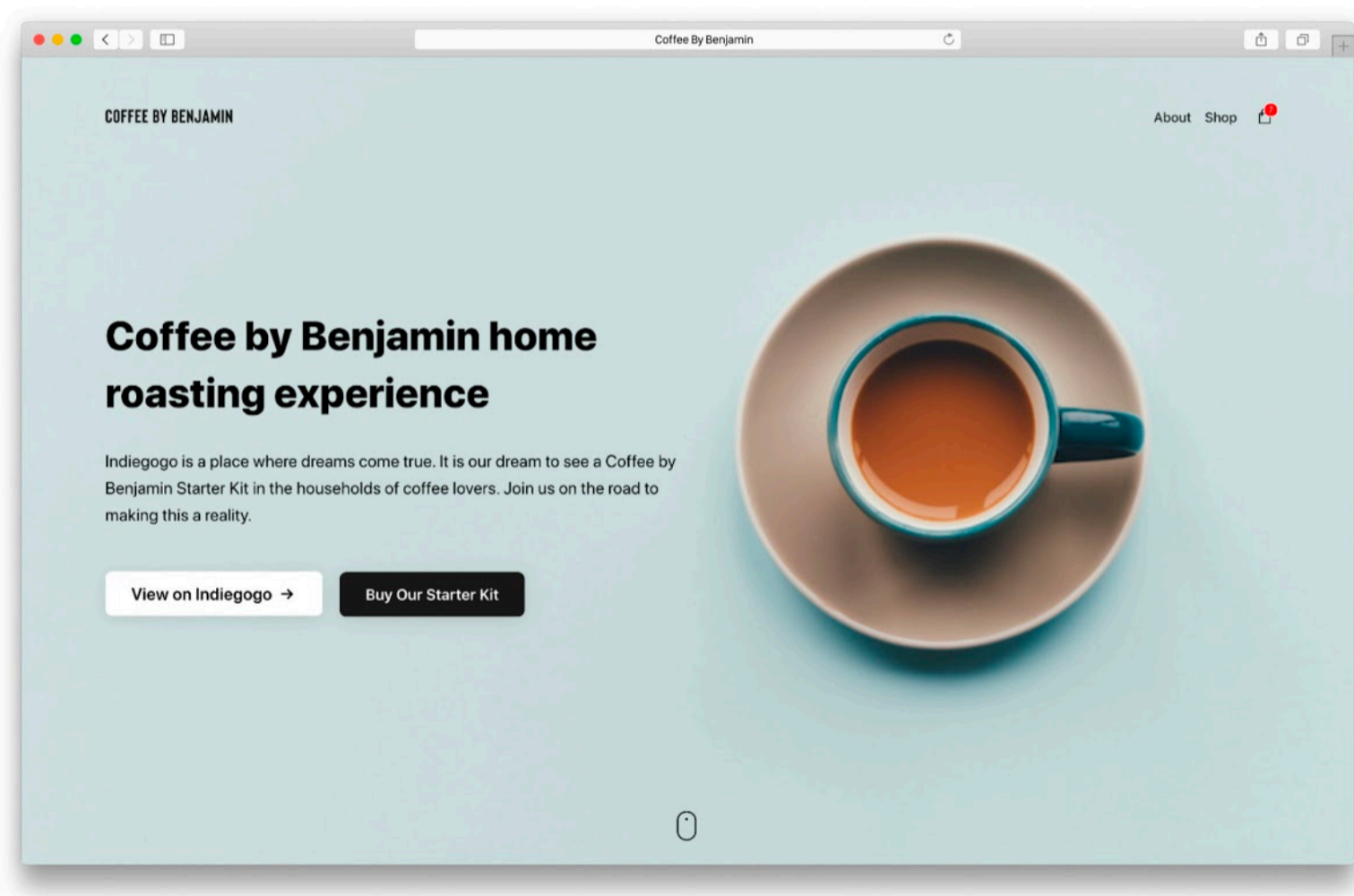
This is a great homepage, but the portfolio *really* shines when you click on one of the listed projects. Here are some screenshots from the second listed project:

JU
LIA.

Coffee By Benjamin

Coffee by Benjamin is a React Application built for a self roasting coffee kit. I built this project from scratch alongside a designer with React, GraphQL, and Shopify. This e-commerce application required a lot of heavy lifting to create a universal cart and overall shopping experience as well as introduce the product and include support pages.

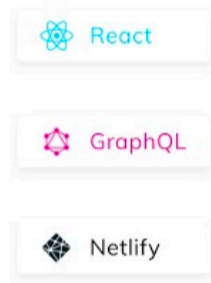
TYPE	STACK	LIVE
Internship	React GraphQL Styled-Components Shopify Netlify	View Site



Project Purpose and Goal

This project included 3 phases and iterations of the site. Phase 1 simply allowed users to enter their email to be alerted to when the product was released. Phase 2 was quite larger and is designed to introduce users to the Coffee By Benjamin product and answer any questions they may have. Phase three is by far the largest and most complex, as it includes the full shop and cart pages as well as the logic and backend that goes along with it.

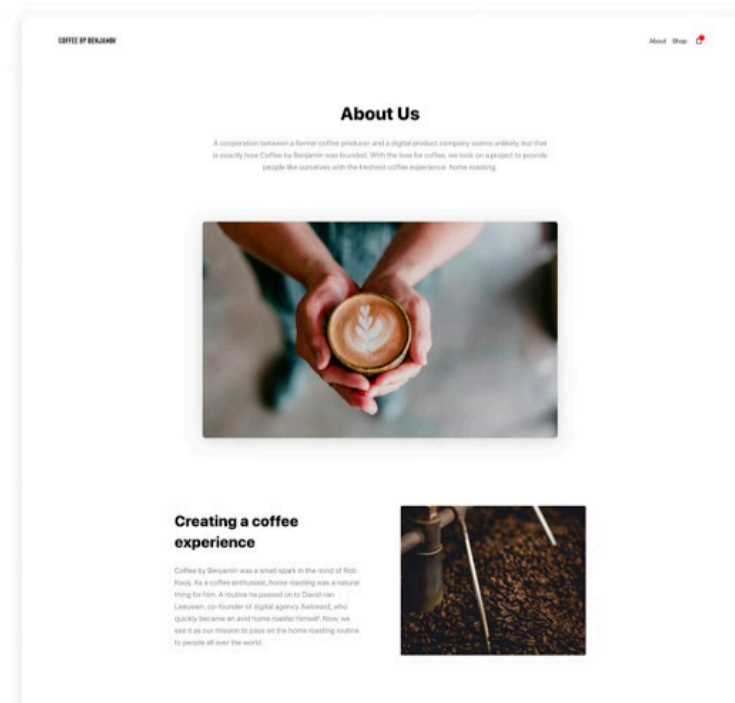
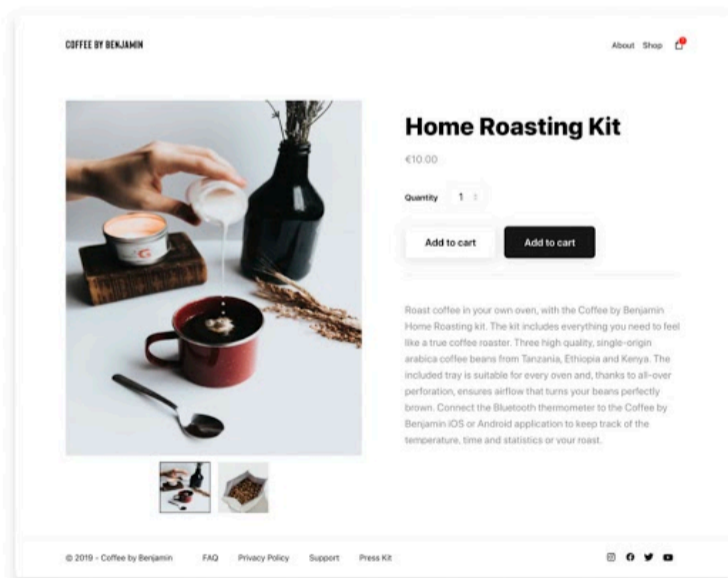
I found that the best way to implement these 3 phases without having separate versions saved was to incorporate a feature flag that will pass the current state that should be displayed and then render content conditionally.

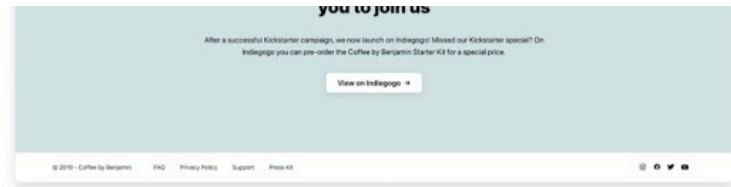
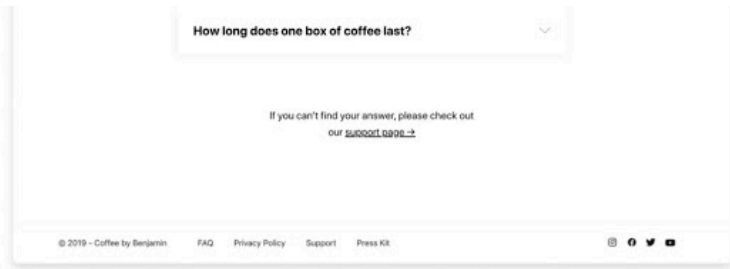


Web Stack and Explanation

React made the most sense for the web application because it required to connect to GraphQL and the Shopify-SDK for javascript ties into React very smoothly. The Shopify-Buy-SDK was chosen because of the ability for the client to modify the products without any complex coding knowledge.

React hooks and session storage are also used throughout the project to maintain the user cart items and allows the cart count and other shopping data to be displayed universally without the need for Redux. Netlify is also an obvious choice for deployment because of its speed and reliability.

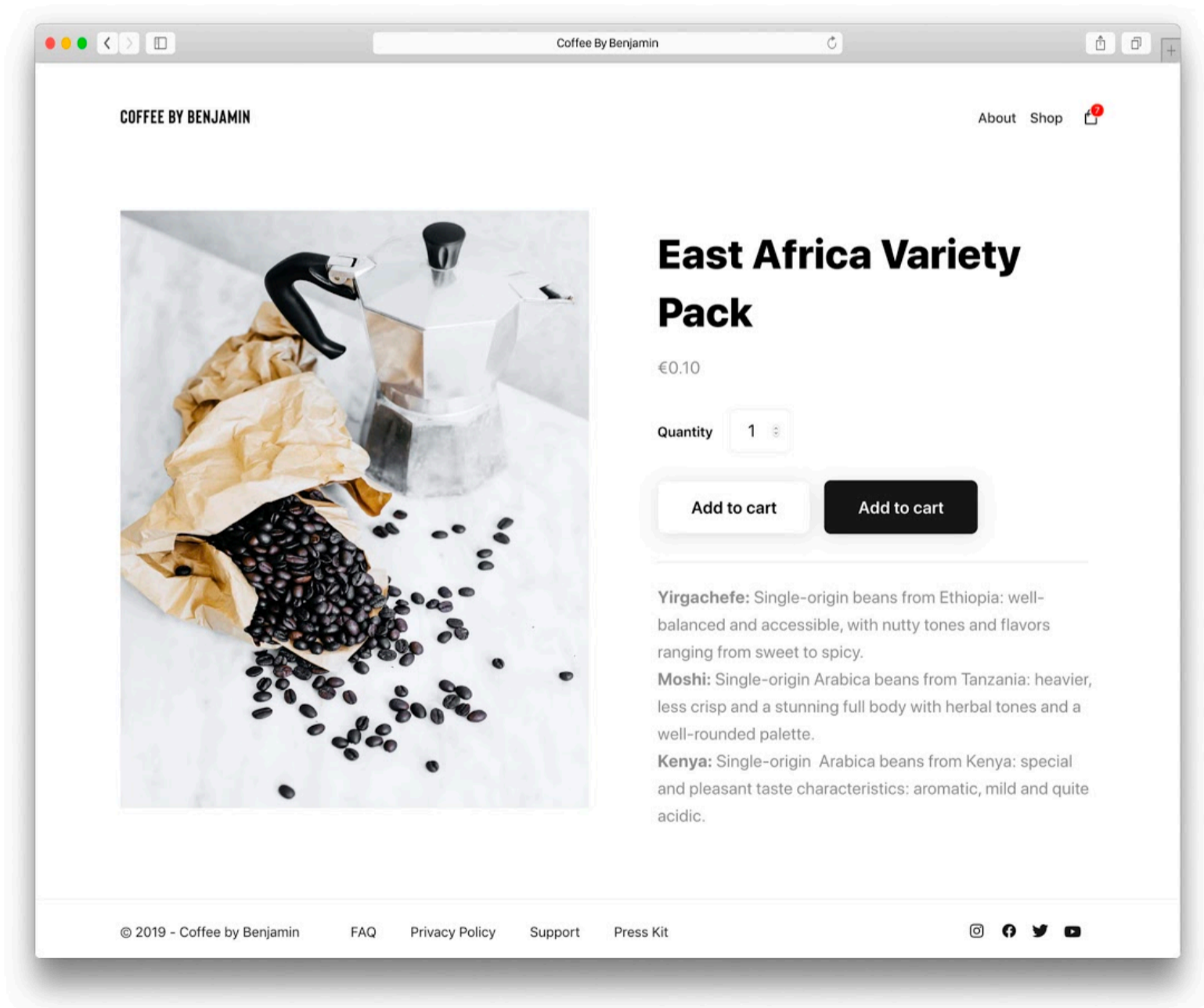




Problems and Thought Process

Like most projects, I ran into a few bumps along the way, one particularly difficult area was organization and structure of the code. Because of this project's size, I realized how important maintaining an organized structure would be.

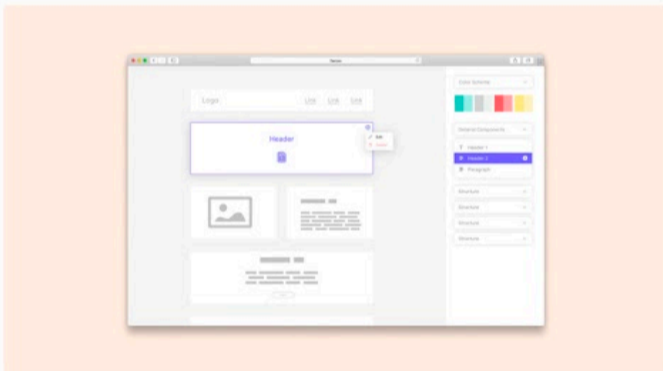
I worked hard to keep components as reusable as possible and utilized props for many slight variations. I also used styled-components, because the structure of CSS-in-js is much clearer and prevents overrides.



Lessons Learned

I could spend all day describing the lessons that I learned while working on this project, but the most important ones involved my newfound understanding of React Hooks, Git management, Feature Keys, and API integration. As my first large project using React, I learned a lot of lessons regarding code structure and organization. When I first began, I would write sloppy code and move on, but now I know how doing so can come back to bite you; I now spend a lot more time refactoring and improving every line I code I write, for the best readability and far fewer headaches.

Other Projects



Decore

A web application that builds custom starter layouts for developers

[VIEW PROJECT](#)

Let's Build Something Together

Feel free to reach out if you're looking for a developer, have a question, or just want to connect.

juliajohnson@purdue.edu

Let's take a look at how this portfolio achieves the attributes we spoke of earlier.

Competence

The project page details a lot of thorny technical challenges, including:

- Using feature flags to incrementally release multiple versions of the product
- Using GraphQL and the Shopify SDK to fetch items from Shopify's headless API
- A focus on reusing code to boost maintainability

You'll notice that Julia never says things like "I'm a fantastic coder" or "I built an amazing project". Julia *shows* rather than *tells*.

Recruiters and HR professionals like to see specific keywords that match the tools and technologies a company uses. Julia lists her skills on the homepage, as well as technologies used for specific projects (in the example we're looking at: React, GraphQL, styled-components, Netlify). She even shares *why* she chose the tools that she did, getting a little extra value out of this section.

Enthusiasm

The impression from reading this site is that Julia really enjoys doing this kind of work – she's taken many courses beyond the content taught in her formal education, and she "always has a project in the works".

There are many ways to demonstrate enthusiasm, but doing it *implicitly* is always better than calling it out *explicitly* ("I am very excited to work as a developer...")

Telling a Story

Julia shares a small amount about her educational background and her priorities, and speaks briefly about the context around the *Coffee by Benjamin* project. She shares the plan of action for developing the application in 3 phases, and what she learned along the way.

I do think there is even more of an opportunity to share some more background. It would be great to hear why she chose to work on a coffee product in the first place, how she collaborated with the designer, and what roadblocks they hit along the way in non-technical

terms (eg. last-minute client requests, design hurdles, etc). Sharing these details can help add a bit of color to the picture, and make the narrative more compelling and memorable.

Humility and Grit

On Julia's homepage, she says:

As I've grown as a developer, I've worked alongside senior designers and developers who have raised my standards for what's expected of any web application.

I love this line. It suggests that her skills are strong for her level, but also that she's easy to mentor. It's exactly the attitude that employers are looking for in junior candidates.

On the project page, Julia mentions that she faced some difficulties with organization and code structure. She was able to overcome these challenges by focusing on creating small, reusable components, and leveraging styled-components. Instead of seeing these challenges as problems, she saw them as learning opportunities; the project summary ends with:

I now spend a lot more time refactoring and improving every line I code I write, for the best readability and far fewer headaches.

Sprinkles

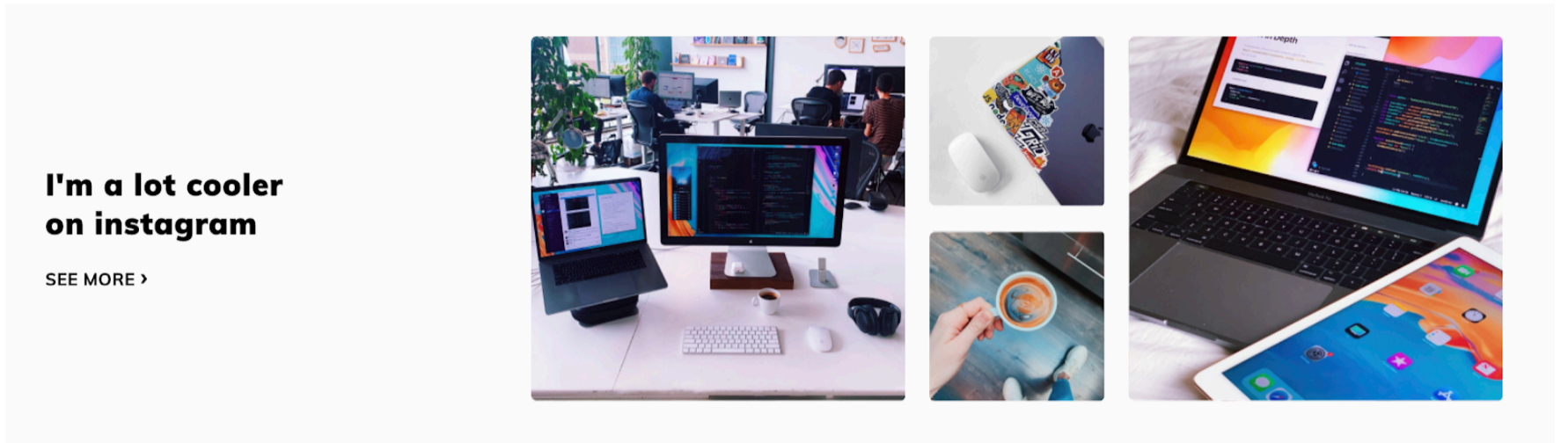
The page features an elegant sequence of on-load animations:

- The main heading fades in, and each line is staggered (They animate in order)
- A line extends, pushing the social media icons out
- The hero image fades in, after a delay

These individual animations work together to create an incredibly polished experience. It's a flourish that captures attention, and makes a sight more memorable.

You can view this animation yourself at <https://www.juliacodes.com/> (though because this isn't my site, I can't guarantee it'll be the same now)

Linking to her personal instagram with a curated selection of photos makes this portfolio unique to Julia, a delightful touch:

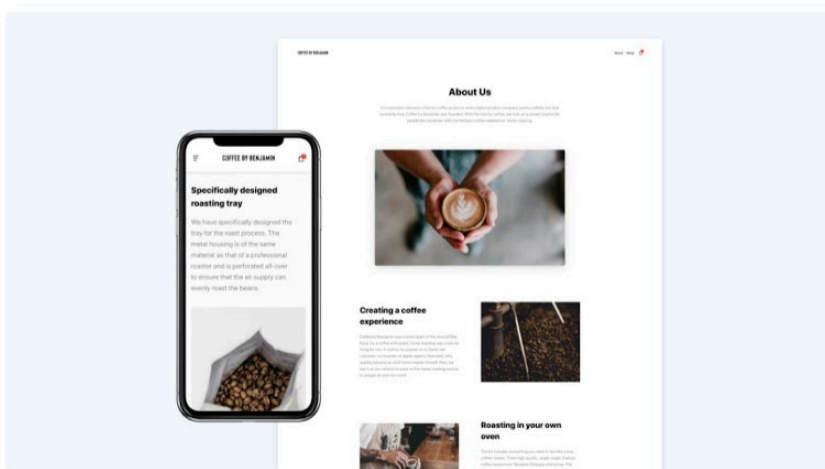


If you pop open the developer console, an easter egg is hidden within:

```
#      #      #####      ###
#      # #     #      #      # ##### ##### ##### ###
#      # #     #      #      # #      #      # #      ###
##### #      #      ##### ##### #      # ##### #
#      # #     #      #      # #      ##### #
#      # #     #      #      # #      #      # #      ###
#      # #     #      #      # ##### #      # ##### ###
```

This website was designed and built by Julia Johnson

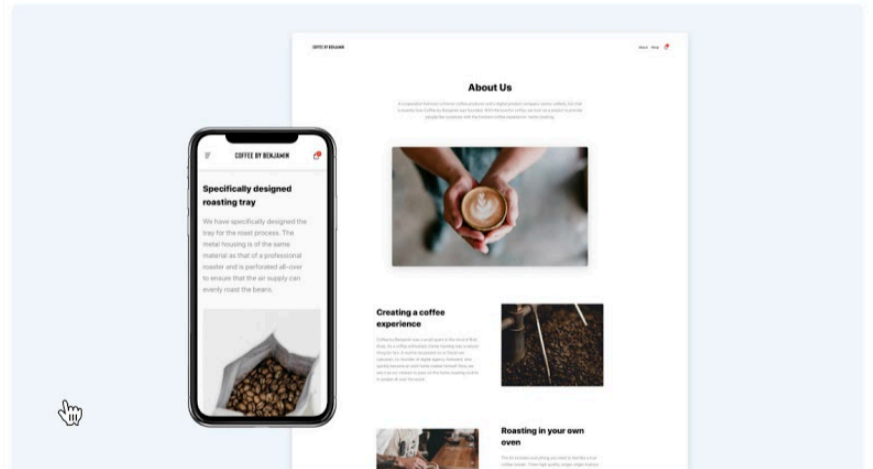
Finally, the website has subtle animations on interactive elements. For example, project thumbnails zoom on hover:



Coffee By Benjamin

Developed a shopify e-commerce application with React for a coffee roasting company

[VIEW PROJECT >](#)



Coffee By Benjamin

Developed a shopify e-commerce application with React for a coffee roasting company

[VIEW PROJECT >](#)

It's hard to tell from these screenshots, but mousing over a project enlarges the image, and adds an underline to "VIEW PROJECT".

IN SUMMARY

Julia's portfolio is a delightful tour through her work, in a way that shines a light on the most impressive aspects. It's a great read.

It's impossible to say how much of a role her portfolio played in her job hunt, but she's clearly been very successful; she received several prestigious internship offers, and currently works at IBM.

Your portfolio should look very different from Julia's—because you're a different person—but it should include all the same signals. It should convey to employers that you're competent and enthusiastic. It should tell a story about your experience and your projects.

In the next chapter, we'll look at the practical things you need to know to build a portfolio site.

CHAPTER IV: BUILDING

We've covered a lot of the high-level theory around what makes an effective developer portfolio. Let's break it down further and talk about how to design and build one.

DESIGN

As a junior or aspiring software developer, does the design of your portfolio matter?

Unfortunately, I think it does. We don't need to have a stunningly-beautiful portfolio, but it needs to look reasonably well-designed.

Even though we aren't applying to work as a UI/UX designer, the people reviewing our portfolios are human, and humans like things that are well-designed. A poorly-designed site can make people feel anxious or uncomfortable, and we don't want to be associated with those emotions. Even if the reviewers make a conscious effort to disregard the design, I suspect it has an effect, and it ultimately colors their impression of us.

The Aesthetic-Usability Effect

*Users are more forgiving of a website with usability problems if it looks pretty. Moreover, this happens subconsciously; people believe that if something **looks** nice, it will **work** well, even if they've experienced usability issues firsthand. This is a well-studied [†] phenomenon.*

*Software development is all about how something **works**, not how something **looks**. The unfortunate reality is that people might judge our software-development skills based on how aesthetic our portfolios site is. Having a nice design can subconsciously suggest that we know how to build good software.*

[†] Source: <https://www.nngroup.com/articles/aesthetic-usability-effect/>

Design is hard. People study for years to become competent designers. You're likely already spending lots of time learning development, and you shouldn't be expected to invest a ton of time into design skills!

So how do we procure a nicely-designed portfolio site without learning all about design? The easiest approach is to copy existing designs. As we've discussed previously, though, we don't want to plagiarize an existing portfolio's design. Instead, I suggest looking for some design *templates*.

At least in my eyes, copying a template doesn't feel as slimy as copying a specific person's template. But we probably shouldn't create a mirror copy. My favourite trick is to find 2-3 sources of inspiration, and merge them into something unique. For example: you can take the layout and spacing from one design, the color scheme from a second design, and an interesting hero detail from a third.

You should always credit the sources that you take inspiration from, unless the inspiration is so subtle that nobody would be able to tell.

Finding a balance

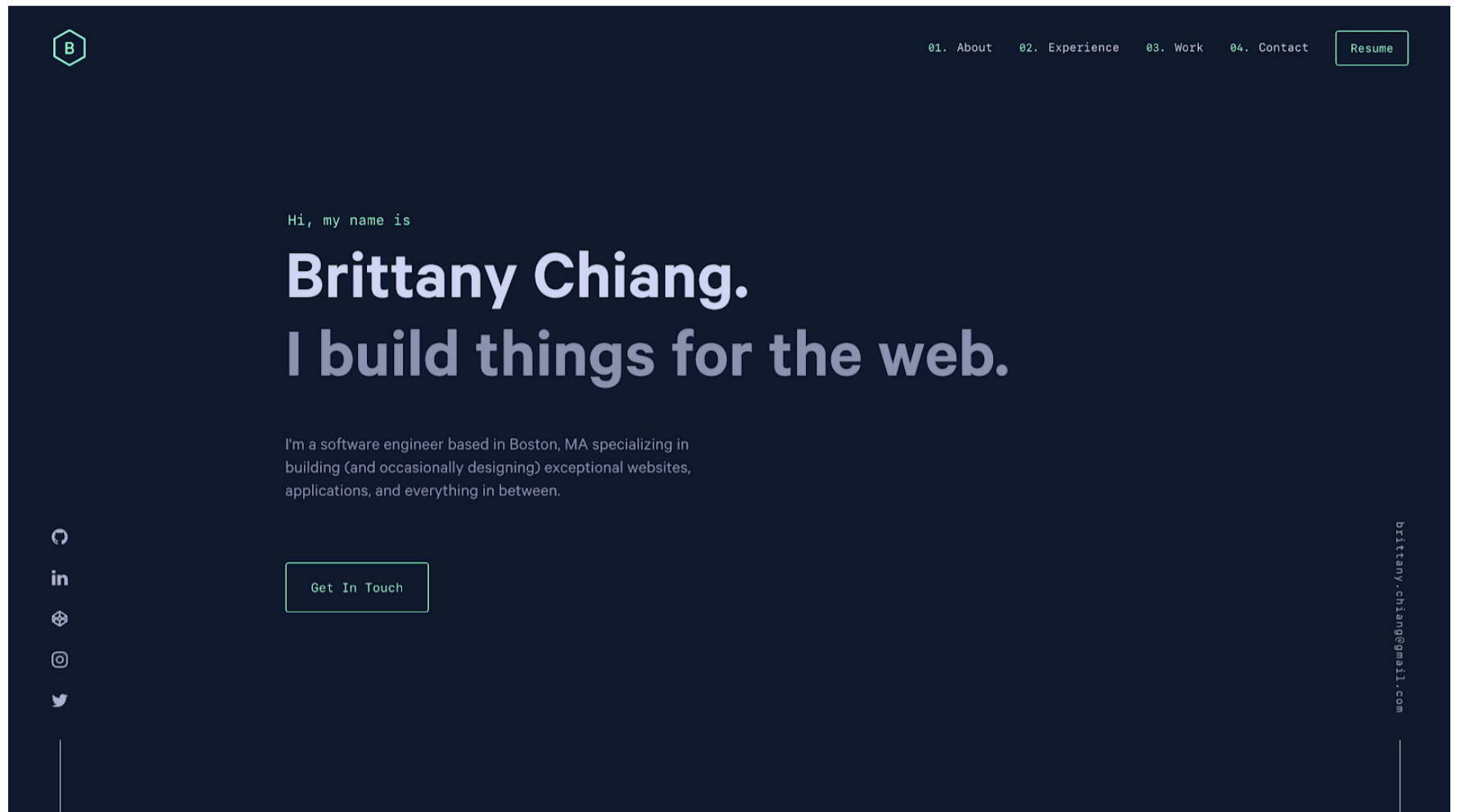
How blatant should you be? My personal standard is that it should be impossible to mistake your site for the thing you're copying from. It's alright if it looks *inspired* by the source, but it should always be clear that *you* built this, and you didn't just copy/paste the HTML and CSS.

The thing is, the further you deviate from an existing design, the more likely it is that you will inadvertently offset what makes the site look professional. This is especially true of spacing; I have no idea why, but even subtle shifts in spacing can break the polished feel of a template. My suggestion is to pick one template you really like and be vigilant about matching the sizes and spacings of elements.

For some example portfolio templates, check out Appendix A. Be creative, as well; feel free to take inspiration from a website you like, even if it's not a developer portfolio.

Crossing the line

One of the most well-known and well-respected portfolio designs comes from Brittany Chiang:



When I was reviewing junior developer portfolios, I saw **several** people steal this design, pixel for pixel.

This design is very unique, and recognizable. I suspect the people who copied it will have a hard time finding work, as recruiters *will* recognize that they're trying to pass off someone else's site as their own.

Please don't do this. Even if the portfolio is open-source, with a permissive license. It leaves a bad taste in the mouth. It's alright to take inspiration, but it should never be so blatant.

FRAMEWORKS AND TOOLING

Once you've conceived of a design, it's time to start building. How should you do it?

There is somewhat of an open debate as to whether developers should code their templates, or if it's acceptable to use a no-code tool like Squarespace or Webflow. My personal opinion

is that if you're seeking a front-end or full-stack role, it's better to create your site with code.

It's likely that many HR professionals won't be able to tell the difference, but developers will know. Whether reasonable or not, some developers will think it's weird that you opted to use a site-builder when you're claiming to have front-end development skills.

I think there is a valid argument that site-builders are quicker to build with, and if you have a rich library of portfolio projects that can demonstrate your front-end skills, perhaps you'll be able to make the case that you were being efficient with your time. But you might not be able to make that case; you just won't hear back, and you'll never know why.

For this reason, I think it's best to create your site from scratch.

Which tools should you use to build your site? There are plenty of options, including:

- Vanilla HTML/CSS/JS
- 11ty (<https://www.11ty.dev/>)
- Gatsby (<https://www.gatsbyjs.com/>)
- Next (<https://nextjs.org/>)
- Jekyll (<https://jekyllrb.com/>)

It's probably best to pick the one you're already comfortable with, since we want to spend our time on the portfolio's substance.

Personally I would pick **Gatsby**. I am biased—I used to work for Gatsby!—but in my opinion it's a fantastic tool for the job. There is a rich library of Gatsby themes and plugins that can reduce the amount of time you have to spend building your site, while still being 100% built by you.

DOMAIN NAMES

If possible, you should buy a domain for your site, *yourname.com*. There are many domain registrars you can use, and registrations typically cost ~\$10 USD per year. I primarily use Google Domains, I've had good experiences with them.

What if yourname.com is already registered by someone else? Here are some suggestions:

- Add a middle initial—my site is JoshWComeau.com
- Use a different TLD (top-level domain, eg. *.com*):
 - *.co* and *.io* are decent generic choices
 - Use one specific to your country (eg. *.ca* in Canada)
 - You can use *.codes* or *.dev* for developer-specific options
- Add a suffix:
 - morgancodes.com
 - dariusdev.com
 - Priyathedeveloper.com

You should avoid the *.info* TLD — it's frequently associated with scams and spam. Also, avoid using a screen-name for the domain (eg. no *laserdude9000.com*), unless the screen-name is reasonably professional and a strong part of your identity.

HOSTING

Your portfolio site should probably be a static site, which means there's no need for a backend server. There are many amazing services you can use for hosting a static site, including:

- Vercel (<https://www.vercel.com/>)
- Netlify (<https://www.netlify.com/>)
- Github Pages (<https://pages.github.com/>)
- Surge (<http://surge.sh/>)

All of these services support custom domains. Research for your specific service to see how to connect them.

You shouldn't need to do anything with services like Heroku or DigitalOcean, nor should you need to pay for monthly hosting with companies like GoDaddy.

PERSONALIZING YOUR PORTFOLIO

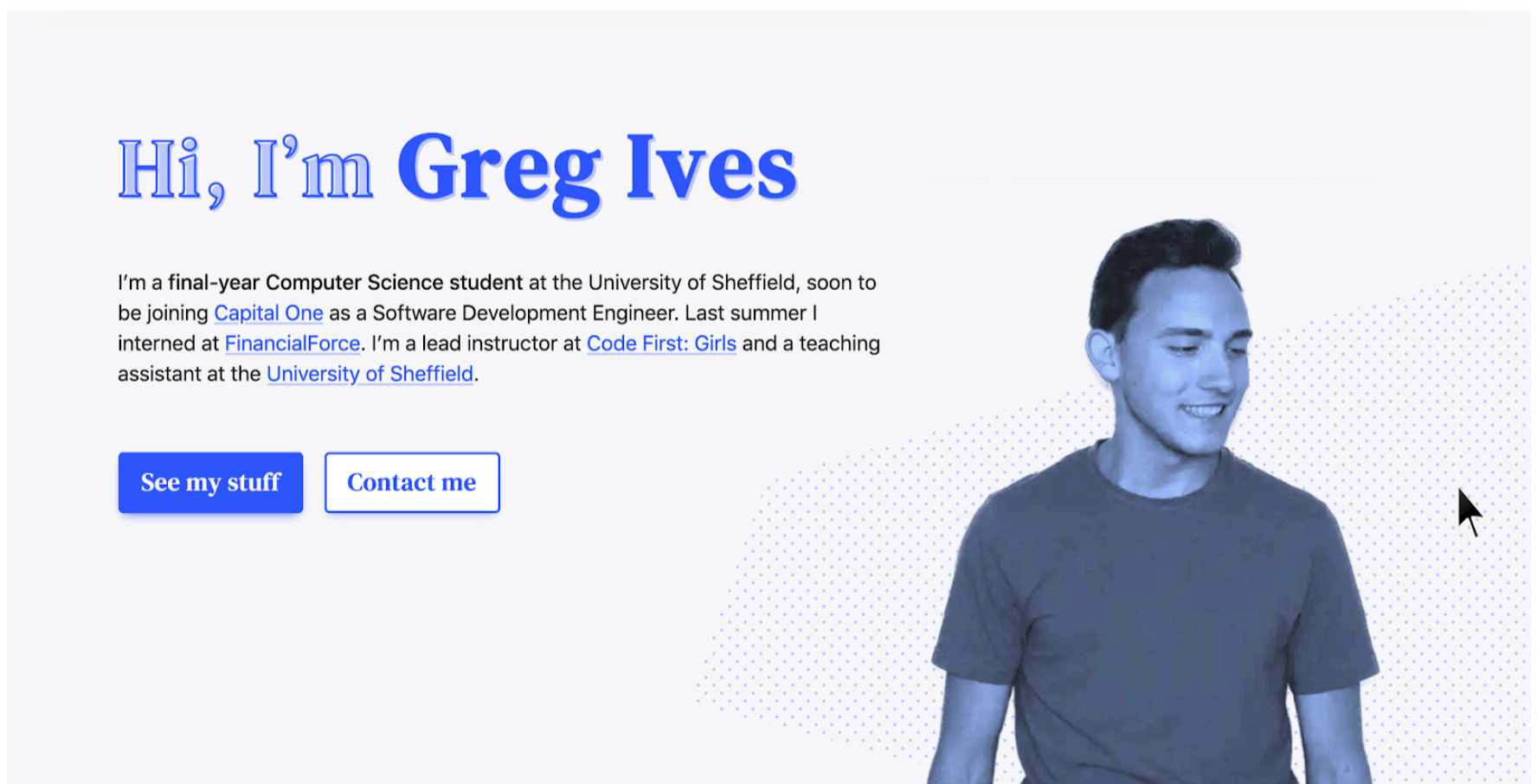
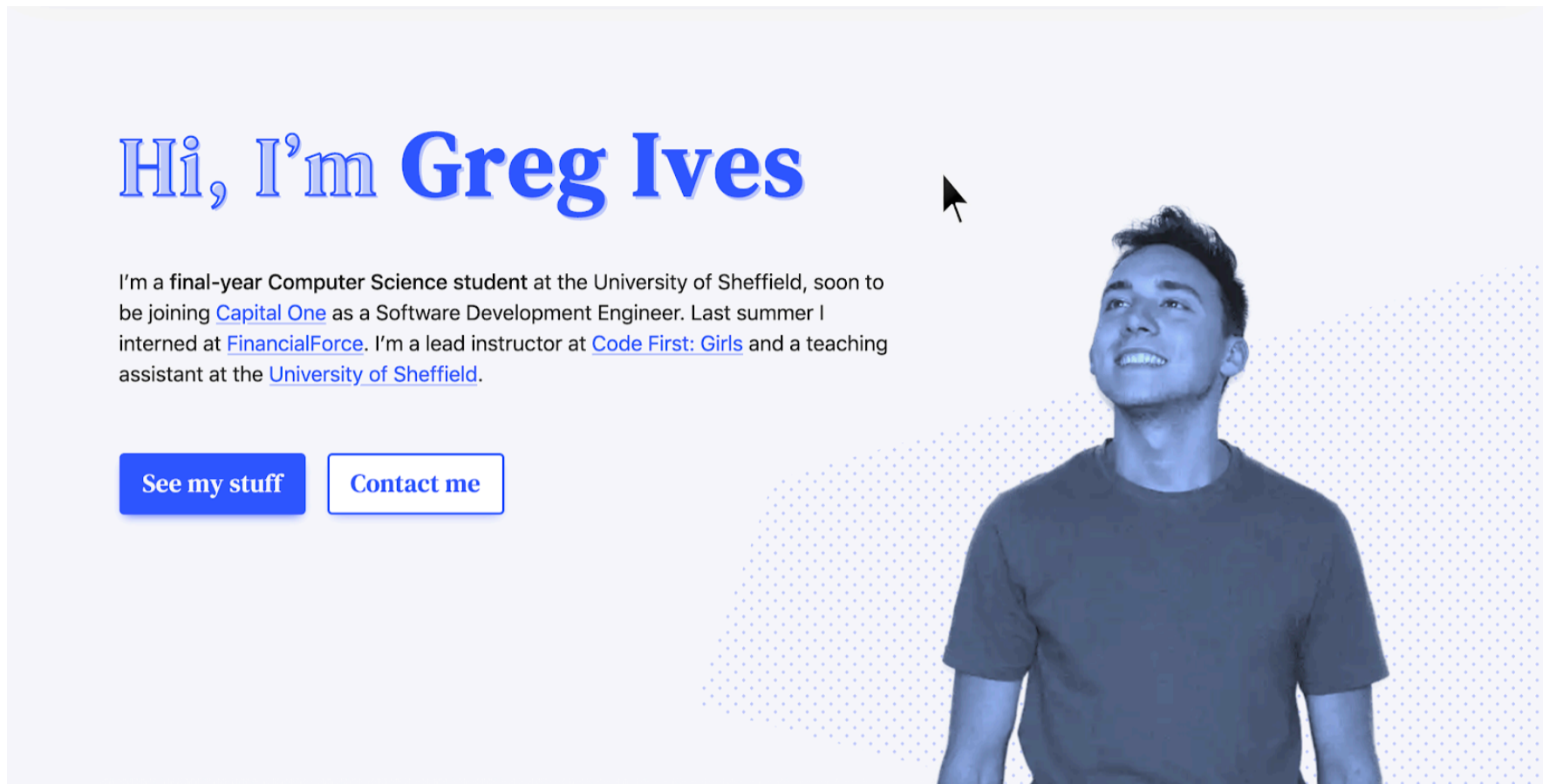
Even though I recommend picking a portfolio template as a starting point, please don't feel like you can't modify it! In fact, you'll likely need to make some modifications; portfolio templates are rarely set up exactly perfectly.

To be clear: our biggest lever for customizing our portfolio will be in the *copy* - how we describe ourselves and our projects. We don't need to make extensive UI customizations.

That said, it can be nice to add some sprinkles, to showcase our enthusiasm for UI development and inject a little whimsy and personality. Some ideas:

- Create a pixelated animated flag for your home country (or LGBTQ+ identity, if applicable)
- Add a dark/light toggle with micro-interactions
- Add a hover animation on interactive elements (links, buttons)
- Add an interaction where it isn't expected

One of my favourite examples of this is from Greg Ives (<https://gregives.co.uk/>). Like many portfolios, it features a photo of himself on the homepage. But as you move the cursor around the page, the photo updates, creating the illusion that he's watching the cursor move:



Cursor tracking is an advanced technique, and this is an exceptional example of a flourish; please don't feel like you have to compete with this. Small, creative details that reflect your personality can go a long way towards making your portfolio stand out and be memorable.

ACCESSIBILITY

It's important that our portfolio site conforms to accessibility standards. Our website should be usable by folks who aren't able to use a mouse and keyboard, as well as blind or colorblind people.

Accessibility is a broad topic, and it's outside of the scope of this book. Here are some resources to help, if you're not already familiar:

<https://a11y.coffee/>

<https://webaim.org/resources/>

<https://websitesetup.org/html-tutorial-beginners/>

You may wonder if this is really a valuable use of time. After all, it's unlikely that the hiring managers or developers will themselves have a disability, right? Do most candidates really submit fully-accessible developer portfolios?

In fact, most candidates don't; many of the junior-developer portfolios I've seen have been fairly inaccessible. But that's all the more reason to excel at this!

Even if the folks reviewing your portfolio don't have a disability themselves, they may know how to test a site to see if it's accessible. This is increasingly becoming a concern to companies, especially in the US, where the Americans with Disabilities Act (ADA) ensures that companies cannot discriminate against people with disabilities. The Supreme Court recently decided, by denying an appeal, that this law applies to websites. Companies are suddenly much more inclined to prioritize developers with accessibility experience.

More information: <https://www.cnbc.com/2019/10/07/dominos-supreme-court.html>

Additionally, it isn't fair to assume that no disabled users will try and access your website; it is estimated that 26% of the United States has at least 1 disability right now, and many will experience a disability, either temporary or permanent, at some point in their lives. All too often, developers neglect this segment of the population, though times are changing.

GENERAL LAYOUT

The design you choose to model your site after will affect the layout, but let's talk about some "must-have" elements. If your template is missing one of these elements, I *thoroughly* recommend adding it.

Your home page should include:

- A brief "About Me" section that highlights your background and personality.
- A list or grid of projects you've worked on.
- Contact information (ideally an embedded form, but a mailto link also works. Don't just include links to social media)

If you're comfortable with the idea, it can also help to add a photo of yourself in the top part of the homepage.

Each project in the list/grid should link to a "Project Details" page, which describes the project in greater depth. This is absolutely critical.

THE "PROJECT DETAIL" PAGE:

Adding "project detail" pages is pretty uncommon among developer portfolios, but it's super common among design portfolios.

The design community has realized that screenshots alone are insufficient. They need to tell a story about their work, to capture all the stuff not shown in the images.

This need is even greater for developers. The work that we do is all about the implementation, not the aesthetics! A screenshot tells very little about our role on the project. And the source code is far too granular, too low-level.

If you only take one thing from this book, make it this: no matter how cool your project is, it cannot be summarized by a screenshot and a short paragraph. Guide the reader through your work!

You might struggle with how to describe this project – filling a page about a project can be a very tall order!

Many junior developers *undervalue* their own projects. It's likely the projects you've built are more impressive than you think! Part of this comes down to marketing.

Years ago, I developed a bit of a reputation amongst friends and family for being good at helping people with their resumes, because I was able to frame their work experience in a positive light. Consider the following excerpt from a resume:

Costco, retail sales, 2018-2020

- Sales associate
- worked on the store floor, Electronics section.

This is an accurate but underwhelming description of the person's experience. I might rephrase as:

Costco, retail sales, 2018-2020

- Developed deep expertise in consumer electronics
- Guided customers towards relevant purchases based on their needs, as well as any relevant promotions and sales
- Operated modern POS systems to check customers out

- Maintained a consistently high approval rating from quarterly guest feedback review cards.

This description is also accurate (assuming that this hypothetical person actually did all of these things), but it sounds much more impressive.

We have a similar goal with our portfolio. We want to be completely honest, but we also want to highlight all of the details that make our project cool. Things that might seem trivial or mundane to you can actually be very impressive to potential employers!

To help get you started, here is a template for what the sections should be, and which information each section should hold.

The exact sections will depend on the nature of your project. Feel free to deviate if it doesn't quite fit your experience.

Introduction

- High-level summary of what the project is
- List of core functionalities / interesting features
- Your role in the project. were you exclusively doing development, or did you do design? If you worked in groups, what parts did you tackle?
- Technologies used
- Links to live demo + source code (if applicable)

Purpose and Goal

- Why did you build this project? Why is it important to you?
- What was the expected outcome of the project?
- What were the initial designs?
- Any other preliminary planning that you did which helps build a narrative

Spotlight

- What is the “killer feature” of your project? What feature does it have that took the most work, or was the most technically impressive? Some possible examples:
 - User authentication
 - A feed of items fetched from a database
 - A particularly tricky UI element (eg. autocomplete, calendar, drag-and-drop)
 - Anything you’re proud of!
- What were the technical hurdles that got in your way? Any major problems you hit during development?
- How did you solve those problems? What was the solution? Go deep here, and write with a developer in mind.

Current status

- This section is optional. *If* the project is actively being used by real people, talk a little bit about the current status, who uses it, why they use it, what they say to you about it, stuff like that.
- If the project was contrived specifically for the portfolio, omit this section.

Lessons Learned

- What did you learn doing this project? Feel free to list multiple things. Also feel free to cover non-technical lessons. It’s great to talk about how you learned to use an advanced feature of a framework or library, but it’s just as valuable to talk about project-management experience, or things you learned about shipping projects.
- If you used a framework or other libraries/tools, was it a good choice? How did it help? In which ways was it insufficient?
- Is your project accessible? What did you learn about accessibility, while building this project? Describing how you tested your project using keyboard navigation or a screen-reader can make for a really compelling story!
- How has this affected the work you’ve done since then? Real examples of how this project built your knowledge for future projects is fantastic.

If you have screenshots of your project (or photos of the process, eg early sketches), you should sprinkle them around this page. Weave them into the narrative, use them to highlight the things you're talking about in the text.

As I mentioned earlier, this template can vary widely depending on your specific project, but this is the level of detail you should aim for. It should take up 2-3 pages when printed (more if you have lots of images).

This is a lot of information, and people often won't spend that long reading. Because of that, **you need to be strategic**: avoid long paragraphs. Make use of many headings, and use lists within those headings to break up chunks of text. In essence, make it **easy to skim**. Some people will get drawn into our narrative and read every word, but we want to make sure the core messages come across even if someone browses quickly.

Video demos?

A trend I've seen occasionally is that developers will include a video walkthrough, showing off all the main features of the project.

This is overall a good idea; it shows that the app is functional, while making sure that a "happy path" is followed. You don't need to worry about the user wandering off and finding a bug, or using a browser that isn't fully supported!

It's not sufficient on its own though. A video will show what the app does, but not things like why you chose to build it, what technical problems you ran into, etc. Plus, videos are harder to skim than text; for that reason alone, many folks will skip watching videos.

If you're particularly proud of specific functionality on your project, I suggest adding a video near the top, but repeating a lot of the information in text below, for folks who skip it.

TONE AND COPY

Finding the right tone for the text you write can be tricky. The most important thing is that you want it to sound natural.

The biggest mistake I see in terms of tone is that developers adopt a “professional” voice that comes across as stuffy and corporate. I tried to model this voice in my “generic portfolio” example.

Why is this such a mistake? Think back to that generic portfolio; even though I asked you to review it carefully, do you remember any of the words from that site?

Formal corporate-speak is incredibly forgettable. It sounds more like an automated telephone robot than a real human. People have learned to tune that tone out, since it doesn't usually carry important information.

Here's your goal: after you've written some text, read out loud, ideally to another person. Does it sound like something you'd say in real life? Rewrite it until it's conversational.

This should be done within reason; if you curse a lot in your day-to-day life, maybe omit that part. But I've almost never seen a developer write *too* casually.

A note on privilege

The copy I write, whether on a portfolio site or a cover letter, tends to be very casual. It has worked well for me, but it would be negligent not to acknowledge the fact that as a white guy, it's likely that my professionalism is assumed. I don't have to prove it in how I write.

In my work as a career coach, I've worked with women and people of color, and I've always been cautious not to assume what worked for me would work for them. Unfortunately, tech is not a meritocracy.

I don't believe that formal corporate-speak is helpful for anyone, but I also recognize that people might want to find a middle ground, and opt for copy that's a little less casual.

For folks who struggle with English.

If English isn't your first language, or if you struggle with grammar and vocabulary, this can be an especially difficult challenge. I would suggest asking for proofreads from friends or peers. If you're not able to find a proofreader,

feel free to shoot me your portfolio site on Twitter (@joshwcomeau). I can't make any promises about my future availability, but I'll do my best to either proofread it myself, or else help you find a proofreader.

DEVELOPER BLOGS

Some portfolio sites feature developer blogs, where developers share tips about what they've learned.

I think that developer blogs are awesome, and I'm a big fan of the idea of learning in public (<https://www.swyx.io/writing/learn-in-public/>). But I'm not sure they should be combined.

As we've discussed, a developer portfolio is built to showcase your best work, and to do it in a very short amount of time. The folks who are looking at your portfolio after you've applied to work at their company likely don't have the time to read any of your blog posts.

I suspect the best strategy is to create an entirely separate blog for yourself (either on a platform like Dev (<https://dev.to/>), or doing it yourself with something like Ghost (<https://ghost.org/>) or Gatsby (<https://www.gatsbyjs.org/>)).

It's unfortunately true that most developer blogs only have one post: "How I built my developer blog". If you make it to your 5th or 6th blog post, you've done something truly uncommon! At that point, I think it'd be a great project to describe in your portfolio.

IN SUMMARY

- Come up with a professional-looking design by copying existing templates. By mixing and matching, we avoid plagiarizing. Whatever you do, don't fork another developer's portfolio and slap your name on it.
- Pick the tools you're already comfortable with to build your site.
- If possible, buy a domain name. You can host your site on free services like Netlify or Vercel.
- Add small flourishes that personalize your site.
- Prioritize accessibility. It's an essential piece of the frontend skillset, and one that employers are increasingly seeking out.
- Your homepage layout should consist of an About section, a Project Details list or grid, and a contact form (or contact details).
- Each project should link to a page that describes it in depth. Use the provided outline if you're not sure what to include.
- If you're interested in creating a developer blog, it should be separated into its own thing. You may wish to list it as a project!

CHAPTER V: PROMOTING YOUR PORTFOLIO

As we've already discussed, not all employers will look at portfolio sites. This chapter include some tips and tricks we can use to increase the number of employers who will see it.

As part of a formal application

Many application forms will ask you for your website. This should absolutely be your portfolio site!

Additionally, there is often a blank text box near the bottom, where applicants can include any information they think is relevant. This is a great place to encourage reviewers to check out your portfolio. You can mention that it includes details on the projects you've built.

Applications will ask you to attach your resume. You can include a link to your portfolio within your resume; I like to include mine by my contact info at the top.

We don't want to overdo it; some employers will never see it, and that's ok. Our goal is to make sure that it's never hard to find, for the employers that are interested.

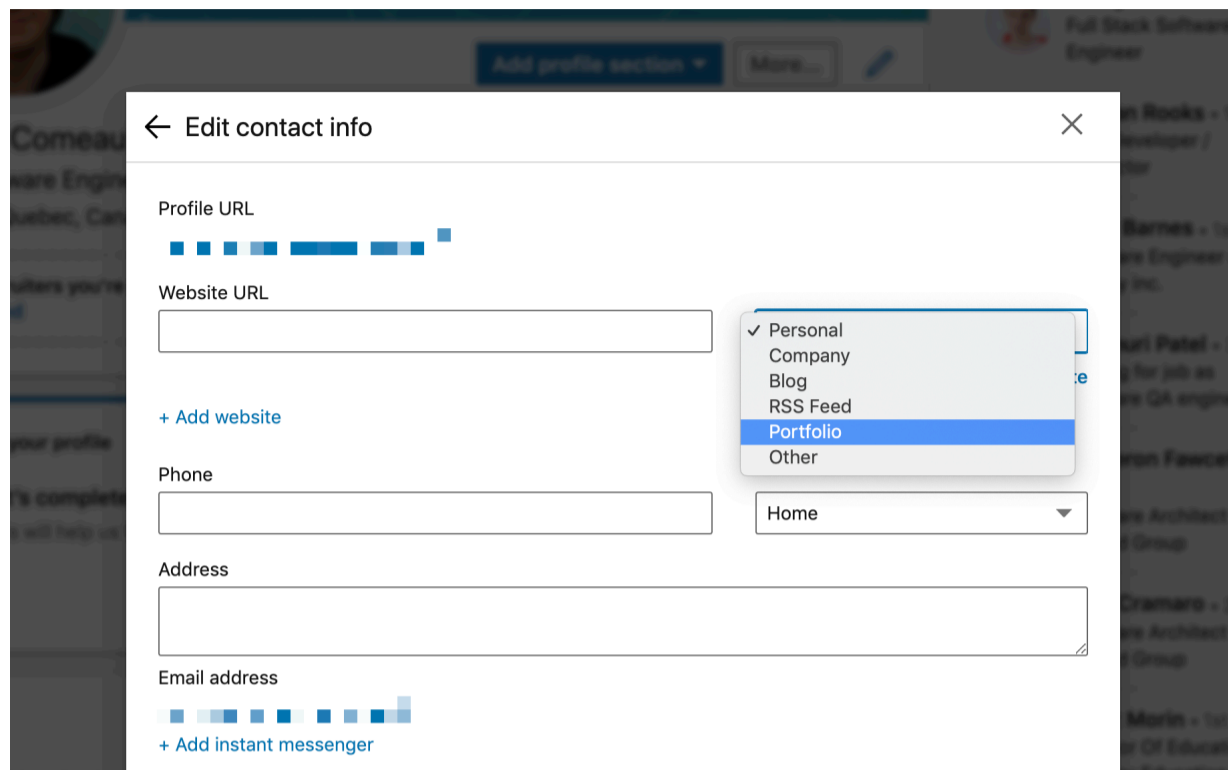
The further you get in your career, the easier it becomes to find work. The early jobs are the hardest ones, because you're competing with so many other aspiring devs for a relatively small number of junior job postings. Having a huge advantage amongst even 10-20% of employers is potentially life-changing. And I suspect far more than 20% of employers will at least glance at your site.

As part of a networking effort

It's estimated that between 70-80% of jobs are never advertised on job boards[†]. Networking unlocks a larger pool of possible jobs. This is often unwelcome news, especially for introverts like me. The silver lining is that it's much easier to show off our stellar portfolio site when we're meeting people through networking.

Networking is far beyond the scope of this book, but here are some quick ways to incorporate your portfolio into these efforts:

- Pin your portfolio site on your Github profile.
- Add your portfolio site to your social media profiles, especially LinkedIn:



- Add a link to your portfolio in your email footer. Every correspondence is a chance for the recipient to discover it.
- Put a link on your business card. I am writing this during the coronavirus pandemic, so this one doesn't feel particularly relevant right now, but once local events come back, this could be worthwhile.
- Work it into conversations, when appropriate. When speaking about your projects, you can say that a bunch more information can be found on your site.

† I found a few sources for this, like this Forbes article (<https://www.forbes.com/sites/jacquelynsmith/2013/04/17/7-things-you-probably-didnt-know-about-your-job-search/#737b18ad3811>) and this LinkedIn study (<https://www.linkedin.com/pulse/new-survey-reveals-85-all-jobs-filled-via-networking-lou-adler/>). Neither of them strike me as undeniably credible. Even if the exact number isn't quite right, the fact remains that there are lots of jobs that go unlisted.

CHAPTER VI: COVER LETTERS

Cover letters are entirely outside the scope of this book, but I'm squeezing it in anyway, because I think it's super important.

Many of the things we've discussed about portfolio sites apply to cover letters. In particular:

- Be conversational. Avoid the empty-calorie "corporate speak".
- Tell a compelling story. Your cover letter should be unique to you. It should feel dishonest if someone else was to copy your letter and replace your name with theirs.
- Keep the HR hiring managers in mind (and to a lesser extent, developers). You're targeting the same demographics.

There's one other cover-letter-specific thing which is super important: **it should be unique for the company you're applying to!**

Far too often, I've seen people copy/paste the same cover letter for each company they apply to, swapping out 1-2 details like the company name and the sector ("I am interested in the _____ industry").

It's OK to copy/paste the parts of the letter that are about your background, but most of the cover letter should be about how your skills and interests overlap with the company's. You should be talking about why you think you'd be a perfect fit for the role. Which specific attributes of the company align with your specific strengths.

As a rule of thumb, 75% of a cover letter should be unique to each company. Don't copy/paste more than 25%.

If you're applying to an agency or a company that doesn't have a specific product, this can be a bit harder, since there isn't an industry you can identify with. But you can learn a lot about a company's culture from their website (and any posted company values), and use that in your letter.

What if you genuinely don't have anything in common with the company you're applying to? Earlier in my career I considered applying to a sports stats site. I don't know anything about sports. This is clearly not ideal, but sometimes we have to settle for unideal companies early in our careers.

In these cases, you can either try to find a tangential interest, or else apply purely based on the culture. You could bend the truth a bit, and pretend to have an overlapping interest, but this is risky; what if someone starts talking to you in the interview about sports?! I'd consider this a last resort.

In Appendix B, I've included a few cover letters I've used which have led to an offer. As with portfolio sites, your cover letter should be unique to you, so please don't plagiarize my letters. But hopefully they can inspire you to write in a similar fashion.

CONCLUSION

I hope this book has helped give you some direction when it comes to your portfolio site!

This book doesn't include any complete examples (aside from the case study), because every person's portfolio site should be unique to them. Remember, the goal is to stand out, not to do what everybody else is doing!

The best portfolio sites are the ones that are able to tell a story about the person and their projects, in a way that makes you feel like you're getting to know the person.

I look forward to seeing what you create—feel free to let me know how it's going on twitter! You can reach me at @joshwcomeau.

A NOTE ON SHARING

If you'd like to share this book with other people, I kindly ask that you direct them to this website: <https://joshwcomeau.com/effective-portfolio/>

This provides valuable analytics about how many people are reading the book, and gives me a way to reach out to readers directly.

This book is and always will be free to download ✨

APPENDIX I: PORTFOLIO TEMPLATES

While I suggest building your own site from scratch, it can be interesting to take inspiration from the templates found on site builders.

We won't take any of the code from these templates, and ideally we won't copy too much from any single one. These are sources of inspiration.

All of these portfolios were meant for designers, not developers. I spent quite a bit of time trying to find developer portfolio templates, and none of them were suitable. Happily, many designer portfolios can be easily repurposed for developers.

This is definitely not a definitive list! Feel free to find inspiration in other places.

- Portfolio Starter — <https://portfolio-starter-template.webflow.io/>

This is a great-looking minimal portfolio, with all the elements we need. Like many of the themes in this appendix, it's targeted at designers, but it works very well for our needs.

- Craig Portfolio — <https://craig-roush-portfolio-template.webflow.io/>

This is a colourful, off-beat theme. I really like it, but it might not be to your taste. It is very recognizable, so you probably don't want to take *too* much inspiration from it.

It has some elements that are more suitable for a freelancer: don't include the testimonials or the “companies worked with” bits.

- Alex Portfolio — <https://alex-grant-template.webflow.io/>

This portfolio has an Apple vibe to it. In terms of layout, it's not great, because it buries the projects. But certain elements are quite nice.

- Dexter Portfolio — <https://dexter-morgan.webflow.io/>

This portfolio is *very* designery. It probably won't be the right fit for most folks, but it might fit with your personality.

- Novo — <https://novo-demo.squarespace.com/?nochrome=true>

This portfolio is very minimal and clean. it's image-heavy, which could be a problem if you don't have a lot of high quality images.

- Kester — <https://kester-demo.squarespace.com/?nochrome=true>

Squarespace has a lot of portfolio templates, and they all kind of look and feel like this. It's nice and minimal, but image-heavy.

- “Art Director” — <https://www.wix.com/website-template/view/html/1913>

This portfolio, like the previous one, depends on having lots of images for your projects. It's probably not the best portfolio template for most folks.

APPENDIX II: COVER LETTERS

This section features cover letters that I've personally used in my applications, and which led to a job offer. I'm including them as an example of the kind of to that can stand out from a crowd, though it's important to point out that my experience is not universal. Adopting a similar tone will not necessarily lead to more job offers for you.

I'm including them to show how personalized they are for the company.

LETTER 1: KHAN ACADEMY

Khan Academy is a non-profit organization dedicated to providing a free, world-class education for anyone, anywhere. Here's what I wrote:

Several years ago, I wanted to build my first web application. I had been doing light web development work (landing pages, contact forms) for a while, and wanted to see if I could build something more interactive.

I had a friend who was working as a tutor, teaching high-school math, and he wanted a way to quiz students online. I've always been passionate about education, and it seemed like an awesome project.

The MVP was very bare-bones, and it was held together with duct tape, but it worked. There was a simple auth system, students would complete quizzes, they would be graded and the results would be emailed to my friend.

The product only worked when you had a tutor available to guide you through the material, though. I wanted to build something that would help students who didn't have access to a tutor. I was exploring the idea of building it into a generalized product with built-in lessons when I discovered Khan Academy.

Khan Academy was almost exactly what I had wanted to build! I went through a few of the lessons, and was thoroughly impressed by the quality of the instructions (Sal Khan has such a talent for teaching) and the clean interface for the practice problems.

Ultimately I realized that I wouldn't be able to build anything that offered a better experience, and have since been building other web applications, but I never stopped being passionate about education, or wanting to build tools to help people learn.

I'd love to chat about why I think I'd be a great addition to the Khan Academy team :)

-Josh

LETTER 2: GLITCH

Glitch is an online code platform designed for education and collaboration. Here's my letter:

Hi Glitch!

My name's Josh, and I'm a senior software developer. I currently work at Khan Academy, but I'm super interested in working at Glitch.

I've been building web things for almost as long as I can remember - I think my first website was a collection of my favourite GIFs when I was 11 or 12, back in the "Welcome to my homepage, it's under construction, sign my guestbook" days. It was rudimentary, but then so was everything else; it did about as much as every other website.

Of course, things have changed a lot since then. I've had the luxury of adapting with the internet, but newcomers nowadays have a much steeper hill to climb. I work part-time at a local coding bootcamp, and it's made me realize just how much stuff you need to know to be able to build something that fits in with today's web.

I think this is a necessary trade-off, because while things are more complicated, the stuff we build is so much more powerful. But I love the idea of making it more approachable. I'm really excited about the work Glitch is doing to enable folks to learn how to build, from the ability to "View Source" across the stack, to the community aspects like being able to request help from others when stuck on a bug, to making source control simple and enabled by default.

I think I'd be a great candidate to help build the community site for a few reasons:

- I have strong technical skills with Javascript and React. Khan Academy is one of the oldest and largest production React applications, and I worked on a project to convert it to a modern single-page-app architecture. I've created and maintained several popular open-source React projects, such as a GUI project manager (Guppy) and an animation library (react-flip-move). I've spoken at several React conferences.*
- I'm passionate about design and UX. I really like building whimsical, delightful experiences, and I believe that it's a huge boost to productivity when you can work closely with design and leverage your own intuition. I do all the design work on my side projects, and it's a ton of fun.*
- I love building quirky, weird, experimental projects. For example, I'm currently working on an app that allows folks to create their own generative art. This gives me empathy for fellow tinkerers, and I'd derive a lot of satisfaction from knowing that my work at Glitch helps folks build similar things.*

Looking forward to hearing from you,

-Josh

LETTER 3: DIGITALOCEAN

DigitalOcean is a cloud computing platform developers can use to host their applications.

Hi there!

My name is Josh, and I'm a senior front-end developer. I currently work for Khan Academy, but I'm super interested in working for DigitalOcean.

I've been a DigitalOcean customer for many years, and I've really enjoyed how accessible it makes server provisioning and administration. As a mostly-front-end developer, DevOps is pretty far outside my wheelhouse, so it's been tremendously useful for me! Empathy is so important for building great products, and I have a built-in empathy for enabling folks to build products they believe in without getting drowned by the complexities of cloud computing.

I think I'd be a great fit for the role for a few reasons:

- I have strong technical skills with Javascript, HTML and CSS. I've been building things for the web in one form or another for over a decade (back when layouts were done with HTML frames and tables), and I've kept up with the industry. For the past few years, I've worked primarily with React.js. I championed the switch from Backbone to React at a previous job, have made significant architectural changes with React at my current job, and have produced a lot of tools for the open-source community, like an animation package and a GUI project manager.*
- I care about the details. I really like working closely with product and design to create amazing, polished user experiences, and will often go the extra mile to include whimsical touches. DigitalOcean's Love is what makes us great value really resonates with me.*

- *I'm passionate about learning! I really liked that the job posting emphasized that DigitalOcean believes in a growth mindset. I'm excited by new challenges and the opportunity to develop further.*

Looking forward to hearing from you,

-Josh